PR-DSig-label-19980403

DSig 1.0 Signature Label Specification

Using PICS 1.1 Labels for Making Signed Assertions

W3C Proposed Recommendation 03-April-1998

Latest Version:
http://www.w3.org/TR/PR-DSig-label
This version:
http://www.w3.org/TR/1998/PR-DSig-label-19980403
Previous version:
http://www.w3.org/TR/PR-DSig-label-971111

Editor

- Philip DesAutels

Authors:

- Yang-hua Chu
- Philip DesAutels
- Brian LaMacchia
- Peter Lipp

Status of this document

This document is in the course of review by the members of the World-Wide Web Consortium. This is a stable document derived from internal Working Drafts of the W3C DSig Working Group and the Proposed Recommendation dated 11 November, 1997. As a result of comments supplied during the first review period, three substantive changes have been made: 1) require support of at least the Digital Signature Standard (DSS) and the RSA+MD5 signature suites for conforming implementations of sigblock 1.0, 2) require support of at least the MD5 and SHA-1 hash algorithms for conforming implementations of resinfo 1.0, and 3) change the URIs of the required DSig 1.0 PICS extension identifiers, resinfo hash algorithm, and signature suite identifiers to conform to W3C documentation standards. The requirement for conforming implementations to provide at least the specified two hash algorithms and signature suites insures a base level of interoperability between all DSig implementations.

Comments on this Proposed Recommendation may be sent to w3c-dsig-ask@w3.org.

At the close of the review period the W3C Director will decide the disposition of this Proposed Recommendation. This specification may become a W3C Recommendation (possibly with minor changes), or it may revert to Working Draft status, or it may be dropped as a W3C work item. This document does not at this time imply any endorsement by the Consortium's staff or member organizations.

This document is part of the W3C (http://www.w3.org/) Digital Signature Project.

A list of current W3C Recommendations, Proposed Recommendations and Working Drafts can be found at: http://www.w3.org/TR

Abstract

The W3C Digital Signature Working Group ("DSig") proposes a standard format for making digitally-signed, machine-readable assertions about a particular information resource. More generally, it is the goal of the DSig project to provide a mechanism to make the statement: *signer* believes *statement* about *information resource*. This document describes a method of utilizing PICS 1.1 labels with extensions to meet this goal.

Contents

DSig 1.0 Overview

The W3C Digital Signature Working Group ("DSig") proposes a standard format for making digitally-signed, machine-readable assertions about a particular information resource. More generally, it is the goal of the DSig project to provide a mechanism to make the statement:

signer believes ***statement*** about ***information resource***

This document describes a method of utilizing PICS 1.1 labels with extensions to meet this goal. It also provides detailed usage guidelines for creating PICS 1.1 labels, which are valid DSig 1.0 Signature labels.

DSig 1.0 signature labels inherit both a means of transporting a signature block data and a simple framework for making the machine-readable assertions from the underlying PICS framework. PICS compliant applications can syntactically parse DSig 1.0 signature labels; only cryptographic functions need to be added to PICS-aware applications in order to make use of the semantic content of a DSig signature.

In its simplest form, a DSig 1.0 signature label is a signed statement about an information resource. This document describes two DSig-specific extensions to standard PICS 1.1 labels: ***resinfo*** and ***sigblock***. The ***resinfo*** extension is used to create cryptographic links between the signature label and the information resource described by the label. Typically this linkage is created through the use of one or more cryptographic hash functions. The ***sigblock*** extension contains one or more digital signatures of the other contents of the label.

In DSig 1.0, it is important to note that:

- At no time does a DSig 1.0 signature label 'wrap' the information resource it is signing;
- The signature label can always be separated from the information resource;
- DSig 1.0 Signature Labels provide a means of making assertions about resources with cryptographic integrity but they do not protect the confidentiality of the information resource referenced.

The basic structure of a PICS 1.1 label is described below.

W3C recommendations and working drafts related to this document include:

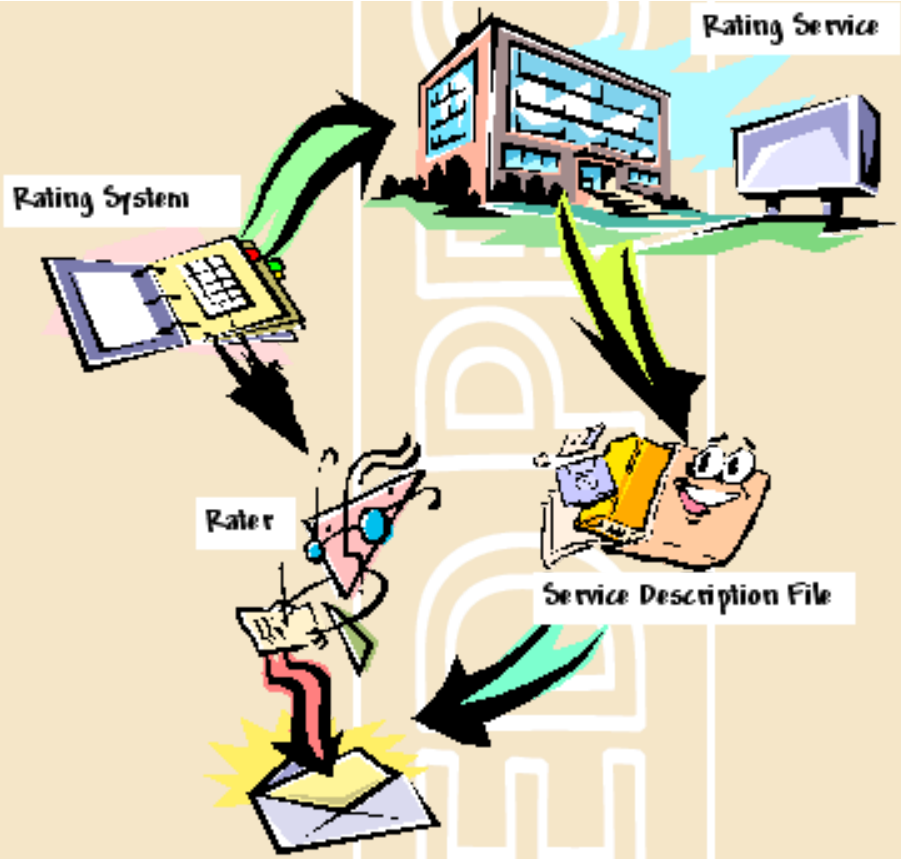
- PICS Label Distribution Label Syntax and Communication Protocols version 1.1: http://www.w3.org/pub/WWW/TR/REC-PICS-labels
- Rating Services and Rating Systems (and Their Machine Readable Descriptions) Version 1.1: http://www.w3.org/pub/WWW/TR/REC-PICS-services

We assume familiarity with these documents.

At the core of DSig 1.0 is the PICS 1.1 label, so we begin by reviewing the PICS 1.1 architecture and illustrating how DSig 1.0 signature labels are built on top of PICS 1.1 labels.

PICS Architecture

At the core of the PICS infrastructure is the rating service. The **rating service** either chooses an existing, or develops a new **rating system** to use in labeling content. The rating system, which is described in a human readable form at the **rating system URL**, specifies the range of statements that can be made. The rating service establishes criteria for determining who can label content using their name and how the labels must be applied. This combination of criteria and rating service are uniquely identified by the particular **service URL**. This service URL becomes the brand, if you will, of the rating service. At a minimum, the service URL will return a human readable form of the rating criteria and a link to the description of the rating system. The rating service is also responsible for delivering a **service description file**. This is a machine-readable version of the rating system with pointers to the rating system URL and the rating service URL. While not required, it is recommended that this be available automatically at the service URL.



A labeler, given authority by the rating service, uses the criteria established by them along with the rating system to label content. These content labels contain a statement about the content of the resource being labeled and contain a link back to the service URL. Content labels can come in the content itself, with the content or from a trusted third party such as a label bureau. Policies determine what actions are taken based on the specific statements in the content label. If a content label is based on an unknown service URL, it is a simple (and potentially automatable) task to retrieve the appropriate service description file to understand what statements are being made in the label.

DSig 1.0 utilizes the PICS infrastructure as described above with a few differences:

- In DSig 1.0, the notion of content labels and raters is somewhat different. In DSig 1.0 the *rater* is considered to be a *labeler*.
- The labeler is making an *assertion* about an information resource and creating an *assertion label*.
- By signing an assertion label, the signer explicitly confirms belief in the truth of the statements contained within the label. A signature does not indicate that the signer created the label, only that they believe the statements made within it are valid."
- Additional signatures can be added in parallel with existing signatures at any point in time.
- The labeler and signer may be, but do not have to be different entities in a DSig label. DSig 1.0 signature labels provide fields for storing information about both the label creator and the signer(s).
- The PICS rating system referenced in the signature label is an *assertion system* in DSig. The statement in the label (made via the PICS ratings) is an assertion that the labeler is making about the referenced content.
- Additional resource reference information may be included within the DSig label to help disambiguate the subject of the label. The DSig ***resinfo*** extension is one way of including such information; it allows the label signer to provide cryptographic hashes of the labeled content. Other private extension types may also be defined and included in accordance with the PICS 1.1 specifications.

PICS 1.1 labels and label lists

PICS labels are always transmitted as lists of one or more individual PICS labels ("label lists"); in common PICS practice PICS label lists usually contain exactly one label. Full details of PICS labels and label lists are available in "PICS Label Distribution Label Syntax and Communication Protocols Version 1.1" document:

- General Format of PICS Labels
- Semantics of PICS Labels and Label Lists
- Requesting Labels Separately

In DSig 1.0, each assertion about an information resource is given in a label. A label consists of a *service identifier*, *options*, *extensions* and an *assertion* (in PICS 1.1 the assertion is called a rating). The service identifier is the URL chosen by the rating service (see Rating Services and Rating Systems) as its unique identifier. Options and extensions give additional properties of the label, the document being labeled and properties of the assertion itself. The assertion itself is a set of attribute-value pairs that describe a document along one or more dimensions. One or more labels may be distributed together as a list which allows for some data aggregation.

A PICS labels contains one or more service sections:

```
(PICS-1.1
   <Service 1 section>
   <Service 2 section>
   <Service 3 section>)
```

Where each service section contains options and labels:

```
<Service URL> <Service options for all labels in this section>
   labels <options for this label> ratings <rating for this label>
          <options for this label> ratings <rating for this label>
          ...
```

The general form for a label list (formatted for presentation, and not showing error status codes) is:

```
(PICS-1.1
   <service 1 url> [service 1 option...]
   labels [label 1 option...] ratings (<category> <value> ...)
          [label 2 option...] ratings (<category> <value> ...)
          ...
   <service 2 url> [service 2 option...]
   labels [label 3 option...] ratings (<category> <value> ...)
          [label 4 option...] ratings (<category> <value> ...)
           ...
   ...)
```

Labels in a label list are grouped by service. Each service may have service options which are inherited by each label within the scope of the service; service options may be overridden by individual label options. When a new service is identified in the label list, the options from the previous service no longer apply. Thus, in the above example label 4 could be equivalently represented as:

```
(PICS-1.1
    <service 2 url>
    labels [ service 2 options + label 4 option...]
           ratings (<category> <value> ...))
```

In DSig 1.0, we sign individual labels or portions thereof; the details of signing labels are presented below.

PICS defines two distinct types of labels, *specific* and *generic*:

- A *specific* label applies to a single resource. For example, if a labeled document is in HTML format, the label applies only to the HTML document itself and not to any other documents referenced via hyperlinks or <img> tags. This is the default label type.
- A *generic* label (identified by the use of the PICS **generic** option within the label) applies to any document with a URL that has a particular prefix (the prefix is specified via the PICS **for** option in the label). A generic label for a site or directory should only be used if it applies to all the documents in that site or directory. The DSig 1.0 ***resinfo*** extension is not meaningful within a generic label.

PICS Options and DSig

Semantics of Embedded Signature Blocks

By convention, a DSig signature block itself has the weakest possible semantics, namely "the entity possessing the key that created this signature had access to the secret key used to generate the signature and the signed data at the same time." For DSig 1.0 signature labels, we want somewhat stronger semantics that also includes the semantics of the ratings contained within the label. Thus, by definition a PICS label which includes a DSig ***sigblock*** extension has the following semantics:

> "The entity possessing the secret key that digitally signed this PICS 1.1 label had access to the secret key and the label at the same time *and asserts that the statements made within the label are valid*"

Applying PICS to DSig

Following the format given in PICS Label Distribution Label Syntax and Communication Protocols Version 1.1 we now review each of the PICS 1.1 options; giving appropriate usage rules for applying them within the context of DSig 1.0 Signature Labels.

PICS label options can be divided into three groups. Options from the first group supply information about the document to which the label applies. Options from the second group supply information about the label itself. Options in the last group provides miscellaneous information.

- **1. Information about the document that is labeled.**

at *quoted-ISO-date*

The last modification date of the information resource to which this assertion applies, at the time the assertion was made . This can serve as a less expensive, but less reliable, alternative to the DSig 1.0 ***resinfo*** extension.

MIC-md5 "*Base64-string*"
-or- md5 "*Base64-string*"

This option is not used in DSig 1.0. If this option is present in a DSig 1.0 label, it should be ignored. Further, it should be removed from the label for the purposes of signing. This option has been superceded by the DSig 1.0 ***resinfo*** extension.

2. Information about the label itself.

by *quotedname*

An identifier for the person or entity who was responsible for creating this particular label. The contents of the by field are not restricted by the DSig 1.0 specification; it is common practice in PICS usage to include a name or e-mail address in the string value of the **by** field. Within DSig 1.0, the **by** field is considered informational only; the **by** option name need not be the same as that of the signer(s). The ***sigblock*** extension includes fields for the identity of the signer (in the *signature* section) and

certificates (or references to them) identifying the signer(s) (within the *attribution information* section).

for *quotedURL*

The URL (or prefix string of a URL) of the information resource to which this assertion applies. This option is required for generic labels and in certain other cases (see "Requesting Labels Separately," PICS Label Distribution Label Syntax and Communication Protocols Version 1.1); it is optional in other cases. The **for** option is valid as both a service and label option in a label list.

generic Boolean

-or- gen *boolean*

If this option is set to true, the label can be applied to any URL starting with the prefix given in the **for** option. By default, this option is false. Set to true, it is used to supply ratings for entire sites or subparts of sites. All generic labels must also include the **for** option. A generic label should not be created unless it can be legitimately applied to *all* documents whose URL begins with the prefix specified in the **for** option (even if a more specific label exists). If the generic option is used with a true value, the DSig 1.0 ***resinfo*** extension can not be used because there will not be a specific information resource to hash.

on *quoted-ISO-date*

The date on which this label was created. This may be different than the date the label was signed (which may be included within the DSig 1.0 ***sigblock*** extension).

signature-RSA-MD5 "*Base64-string*"

This option is not used in DSig 1.0. If this option is present in a DSig 1.0 label it should be ignored and removed from the label for the purposes of signing. This option has been replaced with the DSig 1.0 ***sigblock*** extension.

until *quoted-ISO-date*

-or- exp *quoted-ISO-date*

The date on which the label expires (how long the label is good for).

3. Other information.

comment *quotedname*

Information for humans who may see the label; no associated semantics.

complete-label *quotedURL*

-or- full *quotedURL*

De-referencing this URL returns a complete label that can be used in place of the current one. The complete label has values for as many attributes as possible. This option is used when a short label is transmitted for performance purposes but additional information is also available. When the URL is de-referenced it returns an item of type application/pics-labels that contains a label list with exactly one label. In DSig 1.0 this option might be used if the initial label transmitted was an abbreviated version of the full label. The abbreviated version might contain minimal options and no signature. The client application could then de-reference this URL to get the full, signed version of the label.

extension (optional *quotedURL data**)

-or- extension (mandatory *quotedURL data**)

Future extension mechanism. To avoid duplication of extension names, each extension is identified by a *quotedURL*. The URL is de-referencable, yielding a human-readable description of the extension. If the extension is **optional** then software which does not understand the extension can simply ignore it; if the extension is **mandatory** then software which does not understand the extension should act as though no label had been supplied. Each item of *data* must be one of a fixed set of simple-to-parse data types as specified in the detailed syntax below. See http://www.w3.org/PICS/extensions/ for a partial listing of extension URIs previously defined. The DSig 1.0 ***resinfo*** and ***sigblock*** extensions uses this mechanism (See "DSig Extensions," below for details.)

DSig Extensions

A DSig label 'signs' an information resource. To do this in a secure fashion, the signed label must have a cryptographic connection to that resource. We create cryptographic links between a label and the labeled resource by including one or more hashes of the information resource within the signature label. Similar, albeit limited, functionality was accomplished in the PICS 1.1 specification via the **MIC-md5** (or **md5**) option. DSig 1.0 replaces this option with the ***resinfo*** extension, which permits a single label to include multiple hashes using multiple hash algorithms.

PICS 1.1 also specified a signature option, **signature-RSA-MD5**, but its functionality was similarly limited. DSig replaces **signature-RSA-MD5** with the ***sigblock*** extension. The ***sigblock*** extension may contain one or more signatures using any cryptographic algorithm; in addition, a ***sigblock*** may optionally include information in the form of certificates or links to certificates.

A DSig 1.0 Signature Label is a standard PICS 1.1 label. The DSig extensions ***resinfo*** and ***sigblock*** are both optional and can be used as needed. A PICS 1.1 label is only considered a DSig 1.0 Signature Label when it contains a ***sigblock*** extension.

The syntax of the extensions presented below is written in modified BNF. By convention,

a?
: a or nothing; optional a.

a+
: one or more occurrences of a.

a*
: zero or more occurrences of a.

Quoted strings are case sensitive but other literal elements are case insensitive. Multiple contiguous space characters are be treated as though they were a single space character except in quoted strings.

URLs as identifiers

Within the DSig 1.0 ***sigblock*** and ***resinfo*** extensions, URLs are used as identifiers to indicate a particular hashing algorithm, certificate type or signature type. Specifically, they are used as:

- hash algorithm identifiers
- certificate family identifiers, and
- signature suite identifiers

To ensure the uniqueness of identifiers, the identifier must be a valid URL. This in effect creates a distributed registry of unique names which can be created and shared by any community of interest.

Since the identifier is a URL, it must, when resolved, yield a a document. We recommend the returned document:

- be available at least in HTML format;
- identify the entity which created and maintains the identifier;
- describe the specifics of the algorithms and encodings, or provide a link to another document which does so; and
- be available in multiple languages, either through an existing negotiation mechanism or through links to alternate language versions

We require that such identifier description documents always be provided.

Any incompatible change in an identifier should be accomplished by creating an entirely new identifier URL.

URL identifiers for some common, popular signature suites are available. Of course, DSig 1.0 implementations are not restricted to using these or only these. To provide a base level of interoperability, all DSig 1.0 implementations are required to implement the signature suites listed in Appendix 3.

Resource Reference Information Extension

The goal of the resource reference information (***resinfo***) extension is to provide a cryptographic link between the signature label and an information resource. DSig 1.0's ***resinfo*** extension builds upon the PICS 1.1 **for**, **at** and **MIC-md5** options to provide this cryptographic link. Specifically, the ***resinfo*** extension provides a mechanism for including cryptographic checksums (hashes), in any named cryptographic algorithm, to the label. These hashes provide a means for the receiver of the label to determine if the information resource they have is the same as the one about which the assertion was made.

The ***resinfo*** extension is associated with a specific resource. This resource may be identified by the **for** option or may be implied by the context of the label (in the resource, delivered in the HTTP header with the resource, returned by a label bureau based on a request, etc.).

In the structure of a PICS label, the ***resinfo*** extension can be a *service* option and/or a *label* option. It functions identically to any other option with respect to inheritance within a service section from service option to label option. A single document can have many URLs; the URL used to retrieve a document may differ from the URL in the **for** option of a label that accompanies the document, but the document retrieved must be the same document or the hash(s) contained in the ***resinfo*** extension will not be valid.

Structurally, ***resinfo*** contains one or more hashes of the information resource; each hash includes a hash algorithm identifier, the actual hash of the resource and (optionally) the date the hash was computed.

```
("Hash Algorithm Identifier" "base64-string of hash" "hash date")
```

The hash algorithm identifier is a quoted URL identifier as defined above. It identifies the specific hashing algorithm by which the following hash was computed. The actual hash is given as a quoted base64 encoded string.

Usage notes:

- The ***resinfo*** extension can either be a service option or a label option. If both are present, the extension given as a label option takes precedence over the service option. Thus, an equivalent standalone label will have at most one ***resinfo*** extension and that extension will be the extension given as the label option for that label unless none is present, in which case the extension given as a service option will be used. If neither is present, the equivalent standalone label will have no ***resinfo*** extension.
- A ***resinfo*** extension can contain multiple hashes of the information resource. Each must necessarily use a different hash algorithm; it is not valid to label multiple versions of a single document by including multiple, distinct hashes in one label.
- The ***resinfo*** extension is an "optional" extension. *Optional* implies that even if the processing software does not understand the extension, it should still process the label.
- The ***resinfo*** extension is not valid in a *generic* PICS 1.1 label. It is only valid within a *specific* (non-generic) PICS 1.1 label.

- ***Resinfo*** is not extensible: In DSig 1.0, if other disambiguating or differentiating information is needed, a separate extension will need to be created. We assume that the next version of DSig will allow for much richer and extensible resource reference information.

Detailed Syntax of the *resinfo* Extension in a PICS 1.1 Label

```
resinfo-extension ::= 'extension ( optional '
     '"http://www.w3.org/TR/1998/REC-DSig-label/resinfo-1_0"' resinfo-data+ ')'
resinfo-data      ::= '(' HashAlgoID resource-hash hash-date? ')'
HashAlgoID       ::= quotedURL
quotedURL         ::= '"' URL '"'
resource-hash     ::= '"base64-string"'
hash-date         ::= quoted-PICS-ISO-date
quoted-PICS-ISO-date   ::= '"'YYYY'.'MM'.'DD'T'hh':'mmStz'"'
     based on the PICS-defined ISO 8601:1988 date and time format, restricted
     to the specific form described here:
     YYYY ::= four-digit year
     MM   ::= two-digit month (01=January, etc.)
     DD   ::= two-digit day of month (01 through 31)
     hh   ::= two digits of hour (00 through 23) (am/pm NOT allowed)
     mm   ::= two digits of minute (00 through 59)
     S    ::= sign of time zone offset from UTC ('+' or '-')
     tz   ::= four digit amount of offset from UTC
           (e.g., 1512 means 15 hours and 12 minutes)
     For example, "1994.11.05T08:15-0500" is a valid quoted-PICS-ISO-date
     denoting November 5, 1994, 8:15 am, US Eastern Standard Time
     Notes:
     1. The ISO 8601:1988 date and time format standard allows considerably
     greater flexibility than that described here.  PICS requires precisely
     the syntax described here -- neither the time nor the time zone may
     be omitted, none of the alternate ISO formats are permitted, and
     the punctuation must be as specified here, Except:
     2. The PICS-ISO date described here differs from the ISO 8601:1988
     date and time format. PICS uses '.' as separators while the ISO
     standard calls for '-'. A correct ISO representation for the PICS ISO date
     would be: '"'YYYY'-'MM'-'DD'T'hh':'mmStz'"'. The PICS working group is
     considering extending the PICS recommendation to include support for either
     format. We strongly recommend that DSig implementations support both formats.
     We further recommend that tools creating DSig compliant signature labels use the
     form '"'YYYY'.'MM'.'DD'T'hh':'mmStz'"' until such time as the PICS
     specification is amended to explicitly include support for the correct ISO format.
     If and when this occurs, we recommend that DSig labels be created in the corrected
     format using '-'.
base64-string     ::= as defined in RFC-1521.
URL       ::= as defined in RFC-1738.
```

The following example shows a valid DSig 1.0 ***resinfo*** extension with two hashes of the referenced information resource.

```
   extension
     ( optional "http://www.w3.org/TR/1998/REC-DSig-label/resinfo-1_0"
                ( "http://www.w3.org/TR/1998/REC-DSig-label/SHA1-1_0" "base64 hash" )
                ( "http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0" "base64 hash"
                  "1997.02.05T08:15-0500" ) )
```

In this example, we begin with the ***extension (optional*** tokens which identify this extension as an optional extension to the PICS label within which it is contained. This declaration is followed by a ***URL***, http://www.w3.org/TR/1998/REC-DSig-label/resinfo-1_0, which provides a unique name for the extension. De-referencing the URL provides human readable information on the extension. Finally we have two repeating subsections of the extension, each of which contain hash information. Here again, de-referencing the hash algorithm identifier URL returns a human readable description, this time of the hash algorithm. In our specific example above, the first hash is of type SHA1. This is

followed by the actual hash data and followed by the date the hash was computed. The second clause uses the MD5 hash algorithm.

The Signature Block Extension

The DSig 1.0 Signature Block Extension (***sigblock***) provides cryptographic protection of the DSig 1.0 label by using digital signature techniques. It identifies

- who has signed the information resource,
- which parts of the label were signed (if not the entire label), and
- which algorithms were used to generate the signature, and
- the signature data itself.

The ***sigblock*** extension can also contain certificates that can be used by a trust management system (TMS) to decide if the signature is trustworthy.

Format Specification

A Signature Block consists of

- Attribution Information, and
- one or more Signatures.

Usage notes:

- The ***sigblock*** extension is an "optional" extension. *Optional* implies that even if the processing software does not understand the extension, it should still process the label. We do not require that the extension be understood (mandatory) because the information contained within the label may be useful to applications that cannot understand the signature information. Whether information contained within an unsigned or unverified label should be used is a trust management question.
- The ***sigblock*** extension is valid in both *generic* and *specific* (non-generic) PICS 1.1 labels.
- The ***sigblock*** extension is extensible via signature suites.
- A ***sigblock*** extension is only valid as a label option.

Here is a structural representation of the ***sigblock*** extension:

```
extension
   ( optional "http://www.w3.org/TR/1998/REC-DSig-label/sigblock-1_0"
     <attribution info> <signature>* ) )
```

Attribution Information

The Attribution Information section contains self-verifiable information related to the creation of the digital signature on the label. In particular, cryptographic certificates asserting identity, authorization or other capabilities may be included here. Certificates may be directly embedded within the Attribution Information section of the ***sigblock*** extension, or URLs pointing to certificates may be included. Attribution Information is not required (i.e. this section of the extension may be empty), in which case trust management systems must depend on other information sources when interpreting the label. Furthermore, the information provided herein may or may not be used by the trust management system in processing the label.

Attribution Information supports any certificate format; the types of certificates included will depend on the public key infrastructure used by the application. Certificate format is indicated by

the certificate family identifier, a quoted URL identifier as defined above. This certificate family identifier, when dereferenced, provides information on the format of the data to follow.

None of the information contained within the Attribution Information section is signed by the label's signature because certificates themselves are expected to be self-verifying. (More precisely, none of the information within the entire ***sigblock*** extension, including the Attribution Information section, contributes to the hash of the label that is signed as part of the signature option.) Thus, applications may augment the contents of the Attribution Information section without invalidating the signature on the label (e.g. newly-discovered certificates may be included in the Attribution Information section as they are found, or an expired certificate may be replaced).

Here is a structural representation of the Attribution Information section:

```
( "AttribInfo"
   ( "CertificateFamilyIdentifierURL" "Certificate Data")
   ( "CertificateFamilyIdentifierURL" "Certificate Data")
                  ...)
```

Here is an example Attribution Information section:

```
( "AttribInfo"
   ( "http://www.w3.org/PICS/DSig/X509-1_0" "base64-x.509-cert")
   ( "http://www.w3.org/PICS/DSig/X509-1_0"
     "http://ice-tel-ca/certs/DN/CN=Lipp,O=TU-Graz,OU=IAIK")
   ( "http://www.w3.org/PICS/DSig/pgpcert-1_0" "base64-pgp-signed-key")
   ( "http://www.w3.org/PICS/DSig/pgpcert-1_0"
     "http://pgp.com/certstore/plipp@iaik.tu-graz.ac.at" ) )
```

Signatures

The Signature section of the ***sigblock*** extension contains the actual digital signature data. Each Signature section contains exactly one signature; multiple Signature sections may be included in the ***sigblock*** extension when multiple, parallel signatures are desired. The syntax of the Signature section is:

```
( "Signature" SignatureSuite SigData+)
```

Being crypto-neutral, DSig 1.0 does not prescribe the use of particular algorithms for generating hashes or digital signatures. DSig 1.0 also does not define any particular format for representing cryptographic information in the ***sigblock***. Instead, we introduce the concept of "signature suites," which bundle together certain hashing algorithms, signature algorithms and representation format. Each digital signature includes a signature suite identifier (a quoted URL identifier as defined above) that tells applications how the signature was generated and how it should be parsed.

Each signature suite:

- specifies the algorithms that have been used for creating the signature, and
- defines the content of any subsequent SigData.

Signature suites have complete control over the contents of the SigData immediately following the signature suite URL. The format of this data must satisfy the SigData portion of the BNF; beyond that requirement, the format of the data is governed by the definition given in the signature suite. DSig 1.0 does define two hash algorithms and two signature suites for interoperability; see Appendix 3. Implementations must implement these four algorithms in addition to any others they may wish to define.

Common SigData fields

Although each signature suite is free to specify its own format for signature data (SigData) fields, there are some types of information that are likely to be used by most signature suites. For example, signature suites need to include the actual cryptographic data that constitutes a digital signature. Signature suites will probably also wish to include information about the cryptographic keys used to generate and verify the signature. We now define some common SigData fields and their identifying string tokens ("SigTokenString" in the BNF below). These string tokens are *reserved words* in the sense that any signature suite that uses SigData field identified by these tokens must do so in a manner consistent with this specification.

Keyholder tokens: information about keys related to the signature

Mathematically, a digital signature only cryptographically guarantees that at a particular point in time some process had access to both the signing (secret) key and the text of the signed document. The "Keyholder"-type SigData fields of a signature provides information about the key that was used to create the corresponding signature. The key may be bound to some entity (such as a person, server, or organization) by various certificates. There are four common ways to uniquely specify a particular key; each has its own identifying token:

1. Provide the public key directly ("ByKey");
2. Provide a hash (or fingerprint) of the public key ("ByHash");
3. Provide some *name* that is associated with the public key, such as an X.509 "distinguished name" or the UserID string of a PGP key ("ByName"); or
4. Provide the name of a certifying authority (CA) and information, which identifies the desired key to the CA ("ByCert").

To be useful, the information identifying the signing key will lead the application to corresponding certificates in the Attribution Information section (if any) or provide the starting point for fetching certificates from remote sources.

The following subsections specify the content of the SigInfo fields associated with each of these tokens.

"ByKey"

The token "ByKey" identifies the value that follows as the key that should be used to validate the signature (or sufficient information to generate that key locally).

```
( "ByKey" <Key-Value, SignatureSuite dependent> )
```

The format of the included key necessarily depends on the particular signature suite used and must be specified in the signature suite document. Here is an example use of "ByKey" within the Digital Signature Algorithm (DSA) signature suite:

```
( "ByKey"
   ( "P" "base64-encoded-modulus" )
   ( "Q" "base64-encoded-divisor" )
   ( "G" "base64-encoded-number" )
   ( "Y" "base64-encoded-public-key" ) )
```

"ByHash"

The token "ByHash" identifies the value that follows as the hash of the key that should be used to validate the signature.

```
("ByHash" "base-64-encoded-hash-of-key" )
```

Details on how the hash for a key is generated is a property of individual signature suites.

"ByName"

The token "ByName" identifies the value that follows as a name (or other reference) that may be used to identify the corresponding public key. The name that should be provided depends on the relevant public key infrastructure.

```
( "ByName" "Name-as-string-value" )
```

"ByCert"

The token "ByCert" identifies the value that follows as containing the name of a certifying authority (CA) and the serial number a relevant certificate issued by that CA. The name given for the CA depends on the naming conventions of the relevant public key or certification infrastructure.

```
( "ByCert" ( "CA-Name-as-string-value" <CA-Serial-No.> ) )
```

The "On" token: Time of Signature generation

The token "On" identifies the value that follows as the time the label's signature was generated. (This option is distinct from the PICS 1.1 label option "on" which indicates the time at which the label itself was created.) We recommend using this standard element in all signature suites.

The time that the signature was created is encoded as a *quoted-ISO-Date. The format of a quoted-ISO-Date* is *defined in the PICS 1.1 specifications.*

```
("on" quoted-ISO-Date)
```

Notice that the "on" time is advisory only to applications verifying the digital signature; as this section is part of the entire ***sigblock*** extension it is not cryptographically protected by the signature itself. (The contents of the ***sigblock*** do not contribute to the hash of the label that is signed by the signature.) If a cryptographically-protected date is desired, the correct way to implement it is to include the date within another PICS label extension; that extension may then contribute to the hash of the canonicalized label.

The "include" and "exclude" tokens: modifying the canonicalized form of the label

If an application wishes to transmit both signed and unsigned information in a label the suggested method for doing so is to generate two labels (one signed, one unsigned) and send both labels as a PICS label list. However, some PICS 1.1 protocols, including the protocol for requesting labels from a PICS label bureau, require that exactly one PICS label be returned in response to a request, and thus it may be necessary for a signing application to sign only a subset of a PICS label. If the signature suite permits signatures over partial contents of labels, the "include" and "exclude" tokens provide that functionality:

```
( "exclude" field-list )
( "include" field-list )
```

The "include" and "exclude" SigData fields modify the default behavior of the label canonicalizer. "include" indicates that only the fields listed are included in the signature, "exclude" indicates that all fields are included in the signature except those listed. Before a label is signed, it is put into canonical form; the section "Creating an equivalent standalone label" below describes in detail the canonicalization process. PICS labels have many semantically-equivalent forms yet these forms yield distinct hashes, so it is important that signing and verifying applications canonicalize labels in

the same way. After the equivalent standalone label has been generated following the default canonicalization rules, individual label options may be dropped if an "include" or "exclude" option is present. If an "include" option is present, any field not listed in the field-list is removed from the canonicalized label. If an "exclude" option is present, all fields listed in the field-list are removed from the canonicalized label. At most one "include" or "exclude"; field may appear; it is an error to have both an "include" and an "exclude" option.

The value associated with an "include" or "exclude" option (the "field-list") is a list of label fields to be included or excluded in the canonicalized form. There are three types of fields in PICS 1.1 labels: options, ratings transmit/value pairs, and extensions. The format of a field-list is as follows:

```
field-list      ::= '(' option-list? ratings-list? extension-list? ')'
option-list     ::= '(' "options" <options>* ')'
ratings-list    ::= '(' "ratings" <ratings>* ')'
extension-list  ::= '(' "extensions" <quoted-URL>* ')'
```

A field-list is simply at most, one each of: option-list, ratings-list and extension-list, with their associated data. An option-list is a list of PICS 1.1 label option names (e.g. "for" or "by"). A ratings-list is a list of PICS 1.1 ratings service transmit-names (e.g. "suds" in the example "Good Clean Fun" rating service). An extension-list is a list of quoted-URLs where each quoted-URL uniquely identifies a particular PICS 1.1 label extension.

Note: The "include" and "exclude" SigData types exist in this specification strictly to overcome limitations in PICS 1.1 protocols. W3C's new metadata infrastructure, Resource Description Framework (RDF) should not have these limitations and it is the intent of the DSig working group that "include" and "exclude" not be present in the DSig 2.0 specification, which will build on RDF.

The "SigCrypto" token: signature cryptographic data

The "SigCrypto" token identifies the SigData field that contains the cryptographic data that is the signature itself. The format and contents of this field are entirely specified by particular signature suites.

Hashing

Correct hashing is the key to successful signing. DSig 1.0 therefore specifies how a PICS 1.1 label is converted into a unique, canonicalized form which does not include the ***sigblock*** extension (this process is explained in the Signing section below). This canonicalized label is the input to the signature suite's signature algorithm. The signature algorithm may require or accept other inputs in addition to the contents of the equivalent standalone label. For example, the signature suite may pad the data in a particular way, or mix into the hash of the data information concerning the algorithms used to generate the hash and signature.

Parallel and cascaded signatures

Multiple parallel signatures on the same PICS 1.1 label may be created simply by including several "Signature" fields within the ***sigblock*** extension. Cascaded signatures (signatures on signatures) are not supported within a single DSig signature label. To create a cascaded signature, a DSig signature label may be signed using another DSig signature label.

An example *sigblock* extension:

```
   extension (optional "http://www.w3.org/TR/1998/REC-DSig-label/sigblock-1_0"
      ("AttribInfo"
         ("http://www.w3.org/PICS/DSig/X509-1_0" "base64-x.509-cert")
```

```
         ("http://www.w3.org/PICS/DSig/X509-1_0"
          "http://SomeCa/Certs/ByDN/CN=PeterLipp,O=TU-Graz,OU=IAIK")
         ("http://www.w3.org/PICS/DSig/pgpcert-1_0" "base64-pgp-signed-key")
         ("http://www.w3.org/PICS/DSig/pgpcert-1_0"
          "http://pgp.com/certstore/plipp@iaik.tu-graz.ac.at"))
      ("Signature" "http://www.w3.org/TR/1998/REC-DSig-label/RSA-MD5-1_0"
         ("byKey" (("N" "aba21241241=")
                   ("E" "abcdefghijklmnop=")))
         ("on" "1996.12.02T22:20-0000")
         ("exclude" (("extensions" "http://foo/badextension")))
         ("SigCrypto" "aba1241241=="))
      ("Signature" "http://www.w3.org/TR/1998/REC-DSig-label/DSS-1_0"
         ("ByName" "plipp@iaik.tu-graz.ac.at")
         ("on" "1996.12.02T22:20-0000")
         ("SigCrypto" (("R" "aba124124156")
                       ("S" "casdfkl3r489")))))
```

Detailed Syntax of the *sigblock* Extension in a PICS 1.1 Label

Note: This extension is not a valid PICS 1.1 extension because Base64 encoding uses a '/' which cannot be represented as a PICS 1.1 datum. Nonetheless, since quotedURL (which **does** allow the use of '/') and quotedname (which does not) are indistinguishable at a lexical level (and both are legitimate for use as a PICS 1.1. datum), we believe that all existing PICS parsers will support the grammar presented below.

```
SignatureExtension  ::= 'extension ( optional'
                                  SigBlockURL AttributionInfo Signature* ')'
SigBlockURL         ::= '"http://www.w3.org/TR/1998/REC-DSig-label/sigblock-1_0"'
AttributionInfo     ::= '(' '"AttribInfo"' Certificate* ')'
Certificate         ::= '(' CertificateFamilyID CertificateData ')'
CertificateFamilyID ::= quotedUrl
CertificateData     ::= quotedBase64String | quotedUrl
Signature           ::= '( "Signature"' SignatureSuiteID SigData+ ')'
SigData             ::= '(' SigTokenString SigInfo ')'
SigTokenString      ::= quotedName
SigInfo             ::= SigData | quotedURL | quoted-PICS-ISO-date | quotedBase64String |
                        quotedName | number | '(' SigInfo+ ')'
SignatureSuiteID      ::= quotedUrl
quotedURL           ::= '"' URL '"'
URL                 ::= as defined by RFC-1738.
quotedBase64String  ::= '"' base64String '"'
base64String        ::= as defined in RFC-1521.
alpha               ::= 'A' | .. | 'Z' | 'a' | .. | 'z'
digit               ::= '0' | .. | '9'
quotedName          ::= '"' ( urlChar | ' ')+ '"'
urlChar             ::= alphaNumPM | '.' | '$' | ',' | ';' | ':'
                              | '&' | '=' | '?' | '!' | '*' | '~' | '@'
                              | '#' | '_' | '(' | ')' | '/' | '%' hex hex
                     ; Note: Use the "%" escape technique to insert
                     ; single or double quotation marks within a URL
alphaNumPM          ::= alpha | digit | sign
hex                 ::= digit | 'A' | .. | 'F' | 'a' | .. | 'f'
sign                ::= '+' / '-'
number              ::= [sign]unsignedInt['.' [unsignedInt]]
unsignedInt         ::= digit+
quoted-PICS-ISO-date   ::= '"'YYYY'.'MM'.'DD'T'hh':'mmStz'"'
     based on the PICS-defined ISO 8601:1988 date and time format, restricted
     to the specific form described here:
     YYYY ::= four-digit year
     MM   ::= two-digit month (01=January, etc.)
     DD   ::= two-digit day of month (01 through 31)
     hh   ::= two digits of hour (00 through 23) (am/pm NOT allowed)
     mm   ::= two digits of minute (00 through 59)
```

```
S    ::= sign of time zone offset from UTC ('+' or '-')
tz   ::= four digit amount of offset from UTC
      (e.g., 1512 means 15 hours and 12 minutes)
For example, "1994.11.05T08:15-0500" is a valid quoted-PICS-ISO-date
denoting November 5, 1994, 8:15 am, US Eastern Standard Time
Notes:
1. The ISO 8601:1988 date and time format standard allows considerably
greater flexibility than that described here.  PICS requires precisely
the syntax described here -- neither the time nor the time zone may
be omitted, none of the alternate ISO formats are permitted, and
the punctuation must be as specified here, Except:
2. The PICS-ISO date described here differs from the ISO 8601:1988
date and time format. PICS uses '.' as separators while the ISO
standard calls for '-'. A correct ISO representation for the PICS ISO date
would be: '"'YYYY'-'MM'-'DD'T'hh':'mmStz'"'. The PICS working group is
considering extending the PICS recommendation to include support for either
format. We strongly recommend that DSig implementations support both formats.
We further recommend that tools creating DSig compliant signature labels use the
form '"'YYYY'.'MM'.'DD'T'hh':'mmStz'"' until such time as the PICS
specification is amended to explicitly include support for the correct ISO format.
If and when this occurs, we recommend that DSig labels be created in the corrected
format using '-'.
```

Signing

Since even a single DSig 1.0 signature label must be represented as a PICS 1.1 label list, it is important to understand the structure of such a list. This is explained above in the section PICS 1.1 labels and label lists. Here again is a structural representation of a PICS 1.1 label list:

```
(PICS-1.1
   <service 1 url> [service 1 option...]
   labels [label 1 options...] [label 1 signature]
          ratings (<category> <value> ...)
          [label 2 options...] [label 2 signature]
          ratings (<category> <value> ...)
          ...
   <service 2 url> [service 2 option...]
   labels [label 3 options...] [label 3 signature]
          ratings (<category> <value> ...)
          [label 4 options...] [label 4 signature]
          ratings (<category> <value> ...)
          ...
   ...
)
```

Signing a label

The process for signing a label is fairly straightforward whether the label list containing the label is made up of a single label or a series of labels. First we create an *equivalent standalone label for the label to be signed*. Then the equivalent standalone label is canonicalized (similar to canonicalizing a PICS label for transmission). Finally, a digital signature is generated, inserted into a ***sigblock*** extension, and that extension is placed in the label as a label extension. An equivalent standalone label can have at most one ***resinfo*** extension (which it may inherit from the service options) and one ***sigblock*** extension.

Creating an equivalent standalone label

An equivalent standalone label is a PICS 1.1 label list containing a single label. The single label must be normalized to a form where all options are label options (this includes extension options)

and the ***sigblock*** extension (if present) has been removed. From the example label list above, label 4 could be reduced to the single label:

```
(PICS-1.1
   <service 2 url>
   labels [service 2 option...] OVERRIDDEN BY [label 4 option...]
      ratings ([label 4 ratings ...]))
```

This is not yet an equivalent standalone label. We still need to take into account any modifications denoted by "include" or "exclude" specifiers in the ***sigblock*** extension. (If the signature is being created the application knows which fields it wants to include in or exclude from the equivalent standalone label. The "include" and "exclude" options convey this information to applications trying to verify the signature.) The resulting label list is the equivalent standalone label.

Canonicalization of the equivalent standalone label for signing

- For a given PICS 1.1 label, insert a single space character between any two tokens. PICS 1.1 tokens include left and right parenthesis, symbols, quoted-strings, and numbers. Symbols are case insensitive and converted to lowercases. Tokens in multi-value syntax are considered symbols. Do not insert spaces for either the leading left parenthesis or the the trailing right parenthesis of a PICS 1.1 label list. No carriage-returns or linefeeds are present in a cannonicalized label.
- A normalized DSig-1.0 label consists of three parts in order: the PICS 1.1 header, the options, and the ratings. The header part is the 'pics-1.1' symbol followed by the *serviceURL*.
- The option part is headed by the label keyword "l", followed by a set of PICS-1.1 options, including the extensions. The set of options, including the extensions, are determined by the *option-list* and the *extension-list* fields of the *"exclude*" or the *"include*" option in the signature suite. The options are sorted alphabetically (i.e. lexicographically ordered by the unquoted US ASCII characters) by their shortest names (i.e. use *full* instead of *complete-label*, *exp* instead of *until*). Extension options are sorted first by *"extension"* and then by the *"extension" URL*.
- The rating part is headed by the rating keyword "r", followed by a set of transmit name and value pairs. The set of transmit-name and value pairs are determined by the *"rating-list*" field of the *"include*" or *"exclude*" option in the signature. Transmit names are sorted alphabetically.
- When the client computes the equivalent standalone label format described above, it will use all options available to it: both service and label options. This implies a constraint on the server when it decides what options to include in the transmitted set. The transmitted set must include all options necessary as either service or label options to create the same equivalent standalone label as was signed.

An Example

The following example illustrates a step-by-step process to sign a PICS 1.1 label.

Step 1: creates a PICS 1.1 label

```
(PICS-1.1 "http://www.gcf.org/v2.5"
   by "John Doe"
   labels
      for "http://www.w3.org/PICS/DSig/Overview"
      on "1994.11.05T08:15-0500"
      ratings (suds 0.5 density 0 color 1))
```

Step 2: compute the hashes of the document, create the ***resinfo*** extension, and insert it in the label

```
(PICS-1.1 "http://www.gcf.org/v2.5"
   by "John Doe"
   labels
```

```
        for "http://www.w3.org/PICS/DSig/Overview"
        extension
           (optional "http://www.w3.org/TR/1998/REC-DSig-label/resinfo-1_0"
              ("http://www.w3.org/TR/1998/REC-DSig-label/SHA1-1_0" "aba21241241=")
              ("http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0" "cdc43463463="
               "1997.02.05T08:15-0500"))
        on "1994.11.05T08:15-0500"
        ratings (suds 0.5 density 0 color 1))
```

Step 3: canonicalize the label

(NOTE: Carriage-returns in the above example are for display purposes only. Properly canonicalized, this label would have no carriage-returns.)

```
( PICS-1.1 "http://www.gcf.org/v2.5" l by "John Doe"
  extension ( optional
  "http://www.w3.org/TR/1998/REC-DSig-label/resinfo-1_0" (
  "http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0" "cdc43463463="
  "1997.02.05T08:15-0500" ) ( "http://www.w3.org/TR/1998/REC-DSig-label/SHA1-1_0"
  "aba21241241=" ) ) for "http://www.w3.org/PICS/DSig/Overview"
  on "1994.11.05T08:15-0500" r ( color 1 density 0 suds 0.5 ) )
```

Step 4: sign the canonicalized form of the label and insert it in the label

```
  (PICS-1.1 "http://www.gcf.org/v2.5"
     by "John Doe"
     labels
        for "http://www.w3.org/PICS/DSig/Overview"
        extension
           (optional "http://www.w3.org/TR/1998/REC-DSig-label/resinfo-1_0"
              ("http://www.w3.org/TR/1998/REC-DSig-label/SHA1-1_0" "aba21241241=")
              ("http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0" "cdc43463463="
               "1997.02.05T08:15-0500"))
        extension
           (optional "http://www.w3.org/TR/1998/REC-DSig-label/sigblock-1_0"
              ("AttribInfo"
                 ("http://www.w3.org/PICS/DSig/X509-1_0" "efe64685685=")
                 ("http://www.w3.org/PICS/DSig/X509-1_0"
                  "http://SomeCA/Certs/ByDN/CN=PeterLipp,O=TU-Graz,OU=IAIK")
                 ("http://www.w3.org/PICS/DSig/pgpcert-1_0" "ghg86807807=")
                 ("http://www.w3.org/PICS/DSig/pgpcert-1_0"
                  "http://pgp.com/certstore/plipp@iaik.tu-graz.ac.at"))
              ("Signature" "http://www.w3.org/TR/1998/REC-DSig-label/RSA-MD5-1_0"
                 ("byKey" (("N" "aba212412412=")
                           ("E" "3jdg93fj")))
                 ("on" "1996.12.02T22:20-0000")
                 ("SigCrypto" "3j9fsaJ30SD=")))
        on "1994.11.05T08:15-0500"
        ratings (suds 0.5 density 0 color 1))
```

And now we have a valid DSig-1.0 label.

Signing Notes

While PICS allows labels to be truncated to reduce their size, if this is done in DSig 1.0 after signing, the signature will no longer be valid. An alternative is to distribute an unsigned label with the **complete** option pointing to a full, signed label. Client software in need of a signed label can de-reference the **complete** option's URL to retrieve a complete, signed label.

Appendix 1: Service Resource Information

There is a security hole in the above proposal. The semantics of the assertions (ratings) in a PICS 1.1 label are defined by the rating service, and the only information about the rating service contained within the label itself is the service's URL. Since the human-readable description pointed to by that URL is what defines the rating semantics, it is possible under the current scheme for the semantics of the rating service to change *after* the label has been created without invalidating the label.

If this is a concern, a simple policy in the trust engine that evaluates signatures could be established to require a separate signature label on the service description file.

Appendix 2: Transporting DSig 1.0 Labels

DSig 1.0 labels are PICS 1.1 compliant and thus may be transported in the same way as PICS 1.1 labels. PICS Label Distribution Label Syntax and Communication Protocols Version 1.1 identifies three ways that a PICS label can be transported: In an HTML document, With a document transported via a protocol that uses RFC-822 headers, or Separately from the document.

Labels may also exist on their own, referenced via a URL. When the URL is de-referenced it returns an item of type application/pics-labels that contains a label list.

The specifications for embedding a PICS label in an HTML document are well defined. It is possible to use DSig labels in document other than HTML. To do this, a specification for how the label is embedded in that document type and how the document is normalized for hashing into the label must be created.

Appendix 3: Conforming Implementations

In order to conform with this Recommendation, an implementation must support:

- resinfo 1.0 hash algorithms:
  - MD5 hashing (http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0),
  - SHA1 hashing (http://www.w3.org//TR/1998/REC-DSig-label/SHA1-1_0)
- sigblock 1.0 signature suites:
  - MD5 hashing with RSA crypto (http://www.w3.org/TR/1998/REC-DSig-label/RSA-MD5-1_0),
  - SHA1 hashing with DSA crypto (http://www.w3.org//TR/1998/REC-DSig-label/DSS-1_0) (known as DSS).

Acknowledgments

- John Carbajal carbajal@ibeam.intel.com
- Rosario Gennaro rosario@watson.ibm.com
- Amy Katriel amygk@watson.ibm.com
- Rohit Khare khare@w3.org
- Paul Lambert palamber@us.oracle.com
- Jim Miller jmiller@w3.org
- Hemma Prafullchandra hemma@eng.sun.com
- Rob Price robp@microsoft.com
- Paul Resnick presnick@research.att.com
- Pankaj Rohatgi rohatgi@watson.ibm.com
- Andreas Sterbenz sterbenz@iaik.tu-graz.ac.at

Reference Materials

- N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mai Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, 09/23/1993. ftp://ds.internic.net/rfc/rfc1521.txt
- T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators (URLs)", RFC 1738, 12/20/94. ftp://ds.internic.net/rfc/rfc1738.txt
- Digital Signature Label Architecture, *W3C Working Draft*, http://www.w3.org/pub/WWW/TR/WD-DSIG-label-arch
- Rating Services and Rating Systems and Their Machine Readable Descriptions Version 1.1, *W3C Recommendation,* http://www.w3.org/pub/WWW/TR/REC-PICS-services
- PICS Label Distribution Label Syntax and Communication Protocols Version 1.1, *W3C Recommendation,* http://www.w3.org/pub/WWW/TR/REC-PICS-labels

---

Authors Addresses

Yang-hua Chu
Carnegie Mellon University
Department of Computer Science
Wean Hall 5123
5000 Forbes Avenue
Pittsburgh, PA 15213
Email: yhchu@cs.cmu.edu

Philip DesAutels
Formerly: Project Manager, Technology and Society Domain, W3 Consortium
Current Address:
Senior Principal Architect
MatchLogic, Inc.
400 S. McCaslin Blvd.
Louisville, Colorado 80027
Email: philipd@matchlogic.com

Brian LaMacchia
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
Email: bal@microsoft.com

Peter Lipp
IAIK, University of Technology, Graz
Institute for Applied Information Processing and Communications
Klosterwiesgasse 32/I, A-8010 Graz, Austria
Email: plipp@iaik.tu-graz.ac.at

This version is outdated!For the latest version, please look at https://www.w3.org/TR/REC-DSig-label/.