Message Containers and API Framework

Notices

CONTENTS

Revision History:

Version	Date	Notes on version
0.9a	August 18, 2010	This version is considered ready for implementation with the warning that minor changes may occur prior to final publication.

1 OVERVIEW

Content owners send notices to ISPs, colleges and universities, companies hosting servers and other entities that may make content available in conflict the content owners' rights. The Advanced Copyright Notice System (ACNS)[1] provides a means for notice senders and recipients to communicate using XML data structures, facilitating automation.

Traditionally, notices have been sent via email, and email is still supported. Some desire a web services means of accepting notices. This document describes both email and web services support. Parameters around email, such as the use of the subject line, and around web services are defined in this document.

Messages have always been signed for authentication, but this document offers alternative authentication models and provides for data encryption.

ACNS defines communications between notice senders and recipients. There are more participants in the infrastructure, and there are additional interfaces to support those entities. This document also defines the communications mechanism for them.

1.1 Scope

This document specifies message definitions, security, packaging, and delivery infrastructure. This document does not define the formats for specific messages that will be delivered, and instead points to various other specifications such as the *Automated Content Notice System (ACNS 2.0). Notice Message Containers and API Framework* was originally created for ACNS, but will serve as a general specification for packaging and delivering messages of various types.

Other documents for specific APIs will refer to this document.

This document defines the set of acceptable mechanisms to secure, and package messages; specifically:

- Mechanisms for packaging and delivering messages
- Mechanisms for authentication and security.

All XML constraints and conventions are the same for ACNS 2.0. See ACNS 2.0 documentation MPL-ACNS2 at http://http:ww.movielabs.com/ACNS.

1.2 ACNS Transport Legacy

ACNS is commonly used for sending DMCA and other notices. Many content owners (and representatives) are sending notices with ACNS, and there are many recipients processing ACNS XML. Due to this legacy, mechanisms are in place and cannot be modified. The ACNS 2.0 message structure makes only additions to the ACNS 0.7 specification so content owners may send ACNS 2.0 messages to recipients who have implemented ACNS 0.7 processing without negative consequence. That is, ACNS 0.7 recipients will ignore, or more specifically will not notice, the additional attributes and elements and process the notice as they do currently.

[1] MovieLabs document, MPL-ACNS2C. This may be found at http://www.movielabs.com/ACNS

The backwards compatibility extends to the delivery of ACNS notices. Email is the supported mechanism for delivery. Content owners and their representatives who send notices must continue to support email as a delivery mechanism.

While ACNS 0.7 had not messages from notice recipient to sender, ACNS 2.0 does. This requires new return channels. Email is the legacy mechanism.

Going forward, ACNS sending and receiving parties may agree to use mechanisms other than email. These parties are expected to comply with the Web Services interface defined in this specification. This provides more efficient and secure interfaces than email.

1.3 References

1.3.1 ACNS documents

These can be found at http://http:ww.movielabs.com/ACNS

- *Automated Content Notice System (ACNS 2.0).* Describes the ACNS XML information and usage. Requires version 1.1j or later.
- *ACNS 2.0 Email Delivery, MPL-ACNS2-E,* Version 1.0, February 11, 2009. The predecessor to this document.
- *DRAFT ACNS P2P Profile*—defines usage for P2P notices.
- *DRAFT ACNS Server Profile*—defines usage for user generated content (UGC) sites, Cyberlocker, link sites, etc.
- *ACNS 2.0 Version 1.0 XML schema*

1.3.2 RFCs (Request for Comment)

RFCs are generally available online and are therefore not explicitly referenced here.

1.4 Document Conventions

Tabular element descriptions are of the same form as in *Automated Content Notice System (ACNS 2.0).* Tables can show elements, attributes and simple child elements.

An associated XML schema further defines the XML structure and constraints. Inconsistencies should be directed to the authors at acns <at> movielabs.com.

"Must" and "shall" defines explicit requirements. "May" and "should" describes best or recommended practices.

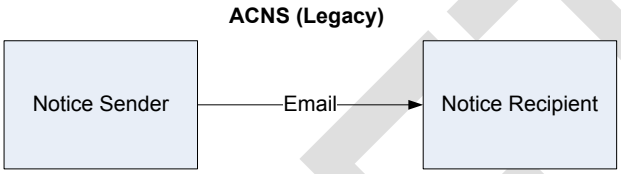
2 MESSAGING MODELS

ACNS originally had messages only from notice sender to recipient. ACNS 2.0 adds messages from notice recipient to sender, so bi-directional messaging is required.

ACNS was originally email-only. For bulk processing of messages, some prefer the use of web services. This document supports models for both email and web services.

2.1 Legacy

ACNS originally supported only email delivery of notices. This model is still supported.



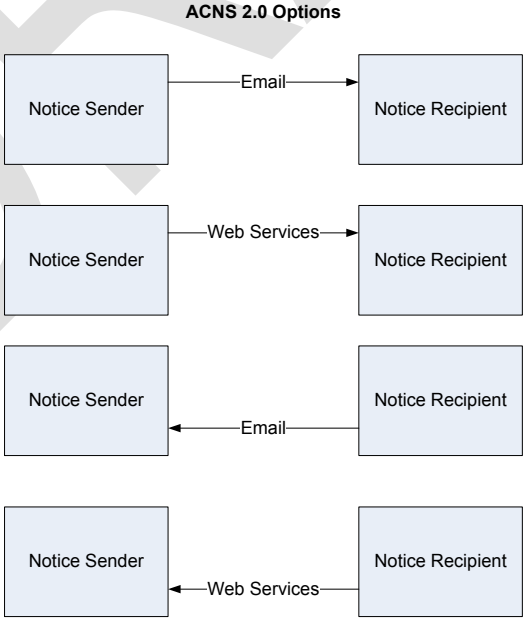
2.2 ACNS 2.0

ACNS 2.0 supports two send and two receive models shown below.

Everyone must be able to send and receive email messages.

Everyone may optionally implement web services. There is no guarantee all other parties will implement web services.

Send and receive need not be symmetric. That is, one may receive messages via web services and return responses via email. However, whatever mechanism is used, it should be used consistently—unless otherwise noted for specific message types.



3 MESSAGE PACKAGING AND SIGNING

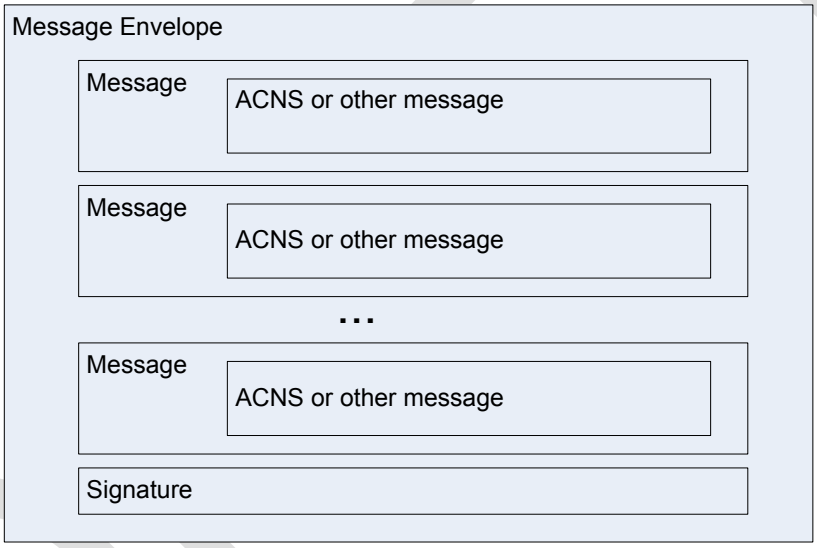
This section defines message containers. These support both web delivery and email delivery.

There are provisions for signing and for packing multiple messages together. Usage of these specific options are defined the delivery mechanisms (i.e., email or web services).

3.1 Note on structure

The message structure needs to provide for multiple message and for signing messages. The Message Envelope is provided to contain multiple messages. If signed, they are all signed with one signature using XMLDSIG as described in *Signed* XML below.

The Message element contains one ACNS or other message element. This container adds a Type attribute to simplify processing upon reception.



3.2 MessageEnvelope element

The MessageEnvelope is the generic container that carries one or more Message elements. It optionally it contains a signature as defined under *Signed XML* below.

When signing, the MessageEnvelope is the XMLDSIG envelope. That is, the signature includes the entire element with the exclusion of the signature child element itself.

The ReplyEmail and ReplyURI attributes in the MessageEnvelope provide information about where reply messages should be sent. These attributes are optional. If ReplyEmail is missing, the reply address is inferred from the ACNS message body. The ReplyURI attribute is populated only if the sender has the ability to receive messages using web services delivery. Note that senders who specify a ReplyURI must be able to accept all messages types that they can receive over the web-based method.

Element	Attribute	Definition	Value
MessageEnvelope			
	ReplyEmail	Email address to send reply messages to (optional)	xs:string
	ReplyURI	URI referential part for reply messages for web-based delivery (optional)	xs:string
	id	ID used for reference from signature. Required if Signature element is included. (optional)	xs:ID
Message		Collection of 1 or more messages (1..n)	See Message Element
Signature		xmldsig Signature. See "Signed XML" below. (optional)	ds:SignatureType

3.3 Message Element

The basic element is the Message element. It contains a single message, such as an ACNS 2.0 NoticeAck. The particular purpose for this element is to include a Type attribute that instructs the recipient the type of message contained, and provide additional information about the message such as a unique message ID, the date and time that the message was created. Note that the sender must guarantee that the message IDs are globally unique. A simple mechanism to do this would be to add the sender's organization's domain address as a prefix or suffix to the message ID.

The Message element contains one of the elements that constitute messages. In ACNS, these are Infringement, NoticeAck, StatusRequest and NoticeStatus. Note that for legacy reasons, in email Infringement is not typically contained in Message element. Other elements can be contained based on usage. Type must match the element name.

Element	Attribute	Definition	Value
Message			
	Type	Type of message: Element name for message type	xs:string "ACNS2.0Notice" "ACNS0.7Notice" "ACNSNoticeAck" "ACNSStatusRequest" "ACNSNoticeStatus"
	ID	Globally unique message id (optional)	xs:string
	Created	Date and Time when the message was created (not necessarily sent or received), (optional)	xs:datetime

(choice of Infringement NoticeAck NotusStatus StatusRequest)		A message element such as NoticeAck, StatusUpdate, or StatusRequest	See acns:Infringement, acns:NoticeAck, acns:NoticeStatus and acns:StatusRequest

3.4 Authentication

All messages must be authenticated. Different delivery mechanisms use different authentication mechanisms (e.g., authenticated secure channels, signed email, and signed XML). In general, mechanisms are discussed in the context of the specific mechanism; for example, email signing is discussed in the email section. General concepts such as trust model and a signed XML container is common to more than one mechanism, and is described here.

3.4.1 Trust

Generally speaking, anyone may send notices to anyone else if they have just cause. Recipients of such notices should be able to determine if the sender is who they say they are to avoid spoofed messages. Standard Internet standards and practices are adopted by this specification. Namely, public key encryption

3.4.2 Signed XML

For message-level authentication, the general process is that the sender generates unsigned messages (based on the appropriate specification for the message), generates a digital signature for that message, and then packages the message with the signature. This package is then sent to the recipient. The signed message contains enough information to validate the sender of the message, and includes both the unsigned message as well as the digital signature of the unsigned message XMLDSIG Signature.

XML Digital Signatures can be used to sign and validate messages across the infrastructure. These shall be in conformance with *XML Signature Syntax and Processing (Second Edition), W3C Recommendation 10 June 2008* as described here: http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/ Note that later versions may be adopted as defined here: http://www.w3.org/TR/xmldsig-core/

The following constraints shall apply when generating digital signatures:

- For CanonicalizationMethod
  - Algorithm=http://www.w3.org/2006/12/xml-c14n11#WithComments
- For SignatureMethod,
  - Algorithm=http://www.w3.org/2000/09/xmldsig#rsa-sha1
- For DigestMethod,
  - Algorithm=http://www.w3.org/2000/09/xmldsig#sha1

A sample XML segment containing a digital signature is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<MessageEnvelope ReplyEmail="example@..." ReplyURI="http://example..." id="envelope"
xsi:schemaLocation="http://www.movielabs.com/ACNS/acnsvx.x.xsd" xmlns="http://www.movielabs.com/ACNS"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Message Created="2001-12-17T09:30:47Z" ID="12345678" Type="ACNSStatusRequest">
...
      </Message>
      <ds:Signature>
            <ds:SignedInfo>
                  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2006/12/xml-c14n11#WithComments"/>
                  <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                  <ds:Reference URI="#envelope">
                        <Transforms>
                              <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </Transforms>
                        <ds:DigestMethod Algorithm="http://www.w3.org/2001/10/xmldsig#sha1"/>
                        <ds:DigestValue>6hpmccmjxQmAI143OhQfIWpkryw=</ds:DigestValue>
                  </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi</ds:SignatureValue>
            <KeyInfo>
                  <X509Data>
                        <X509IssuerSerial>
                              <X509IssuerName>CN=TestSignCert</X509IssuerName>
                              <X509SerialNumber>75496503122422458150193540449068096025</X509SerialNumber>
                        </X509IssuerSerial>
                  </X509Data>
            </KeyInfo>
      </ds:Signature>
</MessageEnvelope>
```

Note that senders must use the same certificate, as defined in the KeyInfo element of the XMLDSig, for all messages using web services. This Key will serve as a unique identifier for the sender, and will be used to describe configuration information (such as URIs) associated with the sender.

Note that the Reference element's URI attribute will always be set to the value "#Body".

The following constraints shall apply when generating digital signatures:

- Data will be transmitted in accordance with section 6.6.4 of that document, "Envelope Transform". XML for encoding may be found here: http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd#enveloped-signature

All web-based delivery mechanisms will support Signed Messages as defined above as a mechanism to sign and validate messages. Email-based delivery will not use XMLDSIG to sign messages.

All recipients of messages must validate Signed Messages before processing them.

Note that all messages require the use of Canonical XML, Version 1.1. (With Comments) http://www.w3.org/TR/xml-c14n11/, which is necessary for proper signing.

4 EMAIL DELIVERY

Email is the default mechanism for communicating messages. All parties should be capable of sending and receiving signed messages via email.

Email messages conform to applicable RFCs (e.g., 2821, 2822, 2045-2049, and others as noted).

4.1 Processing Email Messages

Message sender sends ACNS Email Container (described above) via standard email mechanisms.

Recipient receives email at the specified email address and verifies the authenticity of the message by checking the digital signature. If the received message does not pass the signature test, it may be ignored. Note that this prevents someone from forging notices and flooding a copyright agent. It is strongly encouraged that if this occurs, the sending agent is contacted and warned that someone is forging notices in their name.

If the signature passes the message should be processed.

4.2 Designated Agents

Prior to messages being sent, each potential message recipient provides appropriate email address for delivery of various messages. For example, for ACNS messages, in the United States, a service provider would designate an agent for receipt of DMCA notices as would be found here: http://www.copyright.gov/onlinesp/. Email addresses may be established via separate agreement as appropriate for other types of messages.

4.3 Subject

The subject of the email must start with the message name, followed by optional parameters that are message specific. Message-specific API documents define the format of the subject line for each message type delivered using Email.

The recommended form for a subject line is as follows:

<type> + ": " + <*ID*> + *":"* + <*email*>

Where <type> is the ACNS element type (i.e., "Infringement", "NoticeAck", "StatusUpdate", or "StatusRequest"), <ID> is the "id" element from the "Case" element from the "Infringement" element, and <email> is the "email" element from "Complianant" element from the "Infringement" element.

For example:

```
Subject: Infringement: 123456789:abuse@noticesender.com
```

4.4 Body

The body contains the message. Messages may be signed in accordance with RFC 3156 as discussed in Transport Security.

4.4.1 ACNS Infringement Message ("Notice") Body

The ACNS infringement notice has its own email format for legacy reasons.

The body of an ACNS infringement message contains the cover letter and ACNS XML. Other ACNS messages contain the XML and optionally a cover letter. The entire message must be signed using PGP. This is current practice in ACNS.

Following is an example of an infringement notice:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Dear ISP,

...
Yours Respectfully,
Content Owner.

<?xml version="1.0" encoding="iso-8859-1"?>

<Infringement
xmlns="http://www.movielabs.com/ACNS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
...

</Infringement>
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.8 (FreeBSD)

iQEcBAEBAgAGBQJJbNcWAAoJEL3U2pfJUDus7D0H/im8siM5j7SzlOWkdNL0oTH3
e+hoR4NoOBOelxUkFOHfDZ/ljJJThJJTbeSSryny/VPXwNqo90PNjgsjSy5pYyeC
egccHwSLtE+R6RmqHZn3hmjmoGR7OMXhhRYRWt5acoYuxLak3UL+sPzG69atkLNf
aBEhooIELphfXERn4BjFmksTyZsEfDKEa+iAtoKdIYFG27wegC6RXKQvGDDf/okI
8ZSZiVMCdb/hg+1FSDSdzf3gWNoD9dRy4VK6DYnmSh1Jqw6QjwalaelyAieZZYLe
5mxuCzQIBgknBgdvvkNHMxYIcRrM5LWJDDaCArYz1iWhcf731/oGHq9q/AmPIFc=
=H/9J
-----END PGP SIGNATURE-----
```

4.4.2 Other Message Body

This applies to ACNS messages other than Infringement and all other messages.

The body of the message contains the Message Envelope, describe by the MessageEnvelope element above.

It may optionally contain a cover letter prior to the XML.

The MessageEnvelope must not contain a Signature Element. The entire email body (the cover letter and the MessageEnvelope element) must be PGP signed for delivery.

An example email message body is shown below:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

If you receive this message by mistake, please call
Jonathan Do at 1-310-555-5555.

<?xml version="1.0" encoding="iso-8859-1"?>
<MessageEnvelope>
        <Messages>
            <Message Type=”NoticeAck”>
                …
            </Message>
            …
        </Messages>
</MessageEnvelope>

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.8 (FreeBSD)

iQEcBAEBAgAGBQJJbNcWAAoJEL3U2pfJUDus7D0H/im8siM5j7SzlOWkdNL0oTH3
e+hoR4NoOBOelxUkFOHfDZ/ljJJThJJTbeSSryny/VPXwNqo90PNjgsjSy5pYyeC
egccHwSLtE+R6RmqHZn3hmjmoGR7OMXhhRYRWt5acoYuxLak3UL+sPzG69atkLNf
aBEhooIELphfXERn4BjFmksTyZsEfDKEa+iAtoKdIYFG27wegC6RXKQvGDDf/okI
8ZSZiVMCdb/hg+1FSDSdzf3gWNoD9dRy4VK6DYnmSh1Jqw6QjwalaelyAieZZYLe
5mxuCzQIBgknBgdvvkNHMxYIcRrM5LWJDDaCArYz1iWhcf731/oGHq9q/AmPIFc=
=H/9J
-----END PGP SIGNATURE-----
```

4.5 Message Responses

Some messages require responses to be sent to the sender. Message specific documents outline the response obligations for each message type.

4.6 Payload Encryption

The PGP signed email payload (body) may also be encrypted in accordance with RFC 3156 using the recipient’s public key. This must be by agreement and with appropriate key exchange. Note that RFC 3156 doest not provide for header encryption.

5 RESTFUL WEB SERVICES DELIVERY

Web services delivery is an alternate mechanism for communicating signed messages. If both the sender and the recipient support it, web-based delivery is strongly preferred. If a web-based interface is used for some instances of a message type, it should be used for all messages of that type.

A web services interface includes the following general flow:

- Message recipient provides the associated URIs for the services that will receive different messages
- Vendor generates message XML (based on the appropriate spec) and also generates the digital signature. A MessageEnvelope element is then generated that contains both the Messages element as well as the Signature element, and is used as a post parameter.
- Sender establishes a secure connection with recipient using SSL (TLS 1.1)
- Sender authenticates to recipient using a username/password (HTTP Basic Authentication)
- Sender sends one or more messages.
- Receiver processes the request and validates the message and the sender authentication credentials.
- Receiver then processes the message and immediately returns a status code indicating message has been received and can be processed, or that there is an error and it cannot (an error message is returned).
- Responses are returned as appropriate

5.1 API Style

The core requirement for a web services API is a secure, authenticated channel. Encryption such as TLS 1.1 and authentication such as HTTP Basic Authentication are a minimum expectation. Beyond that, any style of interface is acceptable. Some have chosen to implement as an HTTP POST to a single location; which is fine.

The document defines a RESTful[2] interface[3]. REST, as used, is essentially equivalent to a simple HTTP POST interface for those who have requested such an API.

5.2 Designated Agents

Not every ISP and other recipient will provide a web-based interface. Designated Agent information will be established based on agreements and the results made available to message senders and recipients. Initially, information will be distributed on a case-by-case basis. In the future we anticipate a registration authority.

[2] REST or Representational State Transfer, proposed by Roy Fielding is technique used as part of some web services APIs. See: http://en.wikipedia.org/wiki/Representational_State_Transfer. Services that adopt REST principles are called RESTful.
[3] SOAP may be considered for the future.

Each potential message recipient provides appropriate URI configuration information for various messages that they receive using web-based delivery. Note that receivers must also define the configuration information for email-delivery for use by senders who do not support web-based delivery.

5.3 Transport Security

5.3.1 Encryption

To avoid monitoring traffic, all web service communications will be performed using TLS 1.1 (informally, SSL) in accordance with RFC 4346.

5.3.2 Authentication

The message sender will authenticate the recipient through SSL certificates. Note that the recipient must have a certificate with a chain of trust to a trusted certificate authority (CA). The certificate must match the expected domain that is provided as part of the Designated Agent.

The message receiver must require a login with a secret password known only to the two parties. The username and password will be handled using HTTP Basic Authentication, as per RFC 2617. Sender must use the authentication credentials in the request header for every HTTP request. This is necessary because mechanism is stateless and to avoid Error 401 (“Not Authorized”) redirection.

5.4 Message Envelope

Message may be contained in a MessageEnvelope. If a MessageEnvelope is used, only one XML Document may be included in the MessageEnvelope.

If a Signature Element is used, the signature element must contain the XMLDSIG signature generated using the Messages element as the value and using parameters specified above.

5.5 REST API

This section presents a RESTful API for ACNS. An understanding of RESTful interfaces is assumed.

5.5.1 HTTP and XML

REST APIs use HTTP POST, GET, PUT and DELETE as specified for each API.

HTTP communications in both directions should be conducted with the UTF-8 charset and that the server only needs to support the identity encoding.

All XML exchanged shall be in well formed XML documents.

5.5.2 Status Codes and Error Responses

Status Codes are described in RFC 2616 and are common across all HTTP-based interfaces. (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)

The following HTTP status codes are standard, and should be processed by web services implementers. We do not anticipate other status codes. Behavior specific to notice sending is included below.

- 200 OK. The message was received and is presumed to be processed. No further action required.
- 400 Bad Request: The request was malformed. Specifically, the XML contained does not correspond to a known ACNS or related element. Requests that are properly formed, but cannot be processed for other reasons should result in a response error such as a noticeAck element. An error message is returned.
- 401 Not Authorized: There was an authentication problem due to something wrong with the basic authentication mechanism (e.g., bad or missing username/password).
- 403 Forbidden: System refuses to process a properly formed request. An error message is returned. This occurs when the requestor sends a message that is not allowed; for example, and administrative request from a non-administrator.
- 404 Not Found: An invalid URL or the resource was requested. The domain/path portion of the URL was incorrect, or the REST portion of a URL did not correspond with a valid request type. If the XML can still be processed, a 404 should not be returned.
- 500 Internal Server Error: server side failure.
- 503 Service Unavailable: Service overloaded. Try again later.

As applicable, the error status is returned in the header in accordance with HTTP specification.

REST APIs may additionally return an error message in the body using the RequestError element as defined below:

Element	Attribute	Definition	Type
RequestError			
ErrorNumber		Number of the error	xs:int
Description		Text description of the error	xs:string

5.5.3 REST Notes

The REST resource is a "notice." All methods are in reference to a notice.

Note that the REST API might have been done quite differently had it been the first implementation. However, to support backwards compatibility with existing ACNS delivery based primarily on email, the XML still contains most of actionable information.

5.5.4 REST Endpoint (URL)

HTTP URLs are provided of the form

"https://"+ <recipient hierarchical part> + "/" + <method> [+ <parameters>]

- *<recipient hierarchical part>* is the recipient's domain plus any path they choose for requests.
- The <method> in the URI defines the method being invoked, the value for which is the message name, as defined in the message definition section below.
- <parameters> is the any other portion of the URL

[BaseURL] refers to *"https://"+ <recipient hierarchical part> + "/"*

Different messages will use different parameters to be passed in along with the HTTP request. Message specific API documents will outline the names and acceptable values for all parameters to be used.

5.5.5 NoticeID

These APIs used the concept of noticeID to uniquely identify a notice. This takes the form:

<noticeID> = <ID> + ":" + <email>

Where <ID> is the ID from the Case element and <email> is Email from the Complainant element.

For example, `012345A:abuse@noticsender.com`

5.5.6 ACNS REST Interfaces

5.5.6.1 Notice API

The Notice API is used to POST a new infringement notice.

Path:

`[BaseURL] + "Notice/" + <noticeID>`

HTTP Method:

POST - If new notice

PUT - If the notice is being updated

Request Parameters:

`<noticeID>` is the identifier for the new notice

Request Body:

Body is well formed XML document with an `acns:Infringement` element.

Alternatively, the body may be well formed XML document with `MessageEnvelope` element containing a single `acns:Infringement` element.

Response Body

Valid responses include

- No body, just a status code of 200
- `NoticeAck`
- `NoticeAck` in `MessageEnvelope`
- `RequestError`

5.5.6.2 NoticeStatusRequestID

The NoticeStatusRequestID API is used to request status on an existing notice.

Path:

```
[BaseURL] + "NoticeStatusRequestID/" + <noticeID>
```

HTTP Method: POST

Request Parameters:

`<noticeID>` is the identifier for the notice whose status is requested.

Request Body:

Body is well formed XML document with an `acns:StatusRequest` element.

Alternatively, the body may be well formed XML document with `MessageEnvelope` element containing a single `acns:StatusRequest` element.

Response Body

Valid responses include

- None (successful POST)
- `NoticeStatus`
- `NoticeStatus` in `MessageEnvelope`
- `RequestError`

5.5.6.3 NoticeStatusRequestTimeRange

The NoticeStatusRequestTimeRange API is used to request status on an existing notice.

Path:

[BaseURL] + "NoticeStatusRequestTimeRange/" + <StartDateTime >+"/"+ <EndDateTime>

HTTP Method: POST

Request Parameters:

<StartDateTime> as available in the StartDateTime element of the acns:StatusRequest element in xml datetime format

<EndDateTime> as available in the EndDateTime element of the acns:StatusRequest element in xml datetime format

Request Body:

Body is well formed XML document with an acns:StatusRequest element.

Alternatively, the body may be well formed XML document with MessageEnvelope element containing a single acns:StatusRequest element.

Response Body

Valid responses include

- None (successful POST)
- NoticeStatus
- NoticeStatus in MessageEnvelope
- RequestError

5.5.6.4 NoticeStatusID

The NoticeStatusID API is sent as an asynchronous response to a NoticeStatusRequestID message or it may be sent unsolicited.

Path:

[BaseURL] + "NoticeStatusID/" + <noticeID>

HTTP Method: POST

Request Parameters:

<noticeID> is an identifier for the notice whose notice is being reported.

If this is a response to a NoticeStatusRequestID, then <noticeID> must match the original request.

Request Body:

Body is well formed XML document with an `acns:NoticeStatus` element.

Alternatively, the body may be well formed XML document with `MessageEnvelope` element containing a single `acns:NoticeStatus` element.

Response Body

Valid responses include

- None (successful POST)
- `RequestError`

5.5.6.5 NoticeStatusTimeRange

The NoticeStatusTimeRange API is sent as an asynchronous response to a `NoticeStatusRequestTimeRange` message or it may be sent unsolicited.

Path:

```
[BaseURL] + "NoticeStatusTimeRange/" + <StartDateTime >+"/"+ <EndDateTime>
```

HTTP Method: POST

Request Parameters:

`<StartDateTime>` start datetime of the time range in xml datetime format, and

`<EndDateTime>` end datetime of the time range in xml datetime format

Request Body:

Body is well formed XML document with an `acns:NoticeStatus` element.

Alternatively, the body may be well formed XML document with `MessageEnvelope` element containing a single `acns:NoticeStatus` element.

Response Body

Valid responses include

- None (successful POST)
- `RequestError`

6 LEGACY WEB SERVICES DELIVERY

This section describes a legacy interface for delivering ACNS messages using HTTP POST in a non-RESTful way. This legacy API only supports the delivery of ACNS Notices and is not available for any of the other ACNS messages (such as `NoticeStatusRequest`).

It is strongly recommended that implementers use the RESTful interface defined in section 5 of this document as opposed to this legacy mechanism to deliver messages over HTTP.

6.1 API Style

The core requirement for a web services API is a secure, authenticated channel. Encryption such as TLS 1.1 and authentication such as HTTP Basic Authentication are a minimum expectation. Beyond that, any style of interface is acceptable.

6.2 Designated Agents

Not every ISP and other recipient will provide a web-based interface. Designated Agent information will be established based on agreements and the results made available to message senders and recipients. Initially, information will be distributed on a case-by-case basis. In the future we anticipate a registration authority.

Each potential message recipient provides appropriate URI configuration information for various messages that they receive using web-based delivery. Note that receivers must also define the configuration information for email-delivery for use by senders who do not support web-based delivery.

6.3 Transport Security

6.3.1 Encryption

To avoid monitoring traffic, all web service communications will be performed using TLS 1.1 (informally, SSL) in accordance with RFC 4346.

6.3.2 Authentication

The message sender will authenticate the recipient through SSL certificates. Note that the recipient must have a certificate with a chain of trust to a trusted certificate authority (CA). The certificate must match the expected domain that is provided as part of the Designated Agent.

The message receiver must require a login with a secret password known only to the two parties. The username and password will be handled using HTTP Basic Authentication, as per RFC 2617. Sender must use the authentication credentials in the request header for every HTTP request. This is necessary because mechanism is stateless and to avoid Error 401 (“Not Authorized”) redirection.

6.4 Message Envelope

Message must be contained in a `MessageEnvelope`, and only one XML Document may be included in the `MessageEnvelope`.

If a Signature Element is used, the signature element must contain the XMLDSIG signature generated using the `Messages` element as the value and using parameters specified above.

6.5 HTTP Post

Messages must be sent using HTTP POST with the following parameters:

Type	POST
HTTP POST parameters	*id* parameter containing <noticeID> *messagexml* parameter is MessageEnvelope element with Type= Infringement and Message element containing an ACNS Infringement element.

The following is a sample HTTP POST request and response. Placeholders (highlighted in *green italic*) need to be replaced with actual values.

Note that all of the arguments sent using this method must be URL-encoded.

```
POST /enlighten/calais.asmx/Infringement HTTP/1.1
Host: copyrightcomplaints.isp.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
id=A1234567&MessageXML=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-
8%22%3F%3E++%0D%0A%0D%0A%3CMessageEnvelope%3E%0D%0A%0D%0A++++%3CSignature%3E7
d6cd789c6d87c6d8c76d889d6cd9c7d9c7%3C%2FSignature%3E%0D%0A%0D%0A%3CMessages%3
E%0D%0A%0D%0A%0D%0A%3CMessage+Type%3D%22Notice%22%3E%0D%0A%0D%0A%3CInfringeme
nt++%3E%0D%0A000000000000000000000000000000000000000000%0D%0A%3C%2FInfringemen
t%3E%0D%0A%0D%0A%0D%0A%3C%2FMessage%3E%0D%0A%3C%2FMessages%3E%0D%0A%3C%2FMess
ageEnvelope%3E%0D%0A%0D%0A%0D%0A%0D%0A%0D%0A+
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```