Internet Engineering Task Force	P. Hallam-Baker
Internet-Draft	Comodo Group Inc.
Intended status: Standards Track	May 17, 2013
Expires: November 18, 2013	

TOC

Web Services Connect Protocol draft-hallambaker-wsconnect-00

Abstract

Many Web Services share a requirement for establishing and maintaining a persistent connection between the client and service.

Status of this Memo

Copyright Notice

Table of Contents

1. Definitions
    1.1. Requirements Language
2. Purpose
3. Omnibroker Connection Maintenance Service
    3.1. Authentication
        3.1.1. Broker Authentication
        3.1.2. Device Authentication
        3.1.3. Illustrative example
    3.2. OBPConnection
        3.2.1. Message: Message
        3.2.2. Message: Response
        3.2.3. Message: ErrorResponse
        3.2.4. Message: Request
        3.2.5. Structure: Cryptographic
        3.2.6. Structure: ImageLink
        3.2.7. Structure: Connection
        3.2.8. Bind

3.2.9. Message: BindRequest
3.2.10. Message: BindResponse
3.2.11. Message: OpenRequest
3.2.12. Message: OpenResponse
3.2.13. Message: TicketRequest
3.2.14. Message: TicketResponse
3.2.15. Unbind
3.2.16. Message: UnbindRequest
3.2.17. Message: UnbindResponse
4. Mutual Authentication
4.1. PIN Authentication
4.2. Example: Latin PIN Code
4.3. Example: Cyrillic PIN Code
4.4. Stateless server
4.5. Established Key
4.6. Out of Band Confirmation
5. Acknowledgements
6. Security Considerations
6.1. Denial of Service
6.2. Breach of Trust
6.3. Coercion
7. To do
8. IANA Considerations
9. Normative References
Appendix A. Example Data.
A.1. Ticket A
A.2. Ticket B
§ Author's Address

1. Definitions

TOC

1.1. Requirements Language

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **RFC 2119** [RFC2119].

2. Purpose

TOC

3. Omnibroker Connection Maintenance Service

TOC

3.1. Authentication

TOC

The principle function of the OBP Connection Maintenance Protocol is to establish and maintain the cryptographic parameters used to authenticate and encrypt

The user needs to authenticate the broker service regardless of any authentication requirements of the broker.

3.1.1. Broker Authentication

TOC

The OBP connection maintenance protocol transport MUST provide a trustworthy means of verifying the identity of the broker (e.g. an Extended Validation SSL certificate).

If the device supports a display capability, authentication of the device and user MAY be achieved by means of an authentication image. Such an authentication image is generated by the broker and passed to the client in the OpenResponse message. The user then verifies that the image presented on the device display is the same as that presented on the account maintenance console.

3.1.2. Device Authentication

TOC

If device authentication is required, the mechanism to be used depends on the capabilities of the device, the requirements of the broker and the existing relationship between the user and the broker.

If the device supports some means of data entry, authentication MAY be achieved by entering a passcode into the device that was previously delivered to the user out of band.

The passcode authentication mechanism allows the device to establish a proof that it knows the passcode without disclosing the passcode. This property provides protection against Man-In-The-Middle type disclosure attacks.

3.1.3. Illustrative example

TOC

Alice is an employee of Example Inc. which runs its own local omnibroker service 'example.com'. To configure her machine for use with this service, Alice contacts her network administrator who assigns her the account identifier 'alice' and obtains a PIN number from the service 'Q80370-1RA606-F04B'

Alice enters the values 'alice@example.com' and 'Q80370-1RA606-F04B' into her Omnibroker-enabled Web browser.

The Web browser uses the local DNS to resolve 'example.com' and establishes a HTTPS connection to the specified IP address. The client verifies that the certificate presented has a valid certificate chain to an embedded trust anchor under an appropriate certificate policy (e.g. compliant with Extended Validation Criteria defined by CA-Browser Forum).

Having established an authenticated and encrypted TLS session to the Omnibroker service, the client sends an OpenRequest message to begin the process of mutual authentication. This message specifies the cryptographic parameters supported by the client (Authentication, Encryption) and a nonce value (Challenge), device identification parameters (DeviceID, DeviceURI, DeviceName) and the name of the account being requested.

The client does not specify the PIN code in the request, nor is the request authenticated. Instead the client informs the server that it has a PIN code that can be supplied if necessary.

```
Post / HTTP/1.1
Host: example.com
Cache-Control: no-store
Content-Type: Application/json;charset=UTF-8
Content-Length: 470

{
  "OpenRequest": {
    "Encryption": ["HS256",
      "HS384",
      "HS512",
      "HS256T128"],
    "Authentication": ["A128CBC",
      "A256CBC",
      "A128GCM",
      "A256GCM"],
    "Account": "alice",
    "Domain": "example.com",
    "HavePasscode": true,
    "HaveDisplay": true,
    "Challenge": "d2gdVeQesS3UTOgtK4JSEg==",
    "DeviceID": "Serial:0002212",
    "DeviceURI": "http://comodo.com/dragon/v3.4",
    "DeviceName": "Comodo Dragon"}}
```

The service receives the request. If the request is consistent with the access control policy for the server it returns a reply that specifies the chosen cryptographic parameters (Cryptographic), responds to the client issued by the client to establish server proof of knowsledge of the PIN (ChallengeResponse) and issues a challenge to the client (Challenge).

The cryptographic parameters specify algorithms to be used for encryption and authentication, a shared secret and a ticket value. Note that while the shared secret is exchanged in plaintext form in the HTTP binding, the connection protocol MUST provide encryption.

```
HTTP/1.1 203 Passcode
Content-Type: application/json;charset=UTF-8
Content-Length: 500

{
  "OpenResponse": {
    "Status": 203,
    "StatusDescription": "Passcode",
    "Cryptographic": [{
        "Secret": "11bmdFi9Et7KIUg8aeN2AQ==",
        "Encryption": "A128CBC",
        "Authentication": "HS256",
        "Ticket":
        "TUMnorO0SjHHS7D2uFcGlRYJ0Hd3eibwe0ogptoNMQuCYmCHfHAJcJlyvi
        j8WoXDglTSOkctnmoBzl8W0NLSlcgSyZcmsAyoWs8y1Rn2ZlO2WBgoWrFIO
        qPa4oB29dgs/ei6ieINZtmvXNCm2NUkWA=="}],
    "Challenge": "alX8aAWH6acSqO3FTT94HA==",
    "ChallengeResponse": "enT5myMw8w2hV4H32Ntx/g=="}}
```

To complete the transaction, the client sends a TicketRequest message to the serice containing a response to the PIN challenge sent by the service (ChallengeResponse). The TicketRequest message is authenticated under the shared secret specified in the OpenResponse message.

```
Post / HTTP/1.1
Host: example.com
Cache-Control: no-store
Content-Type: Application/json;charset=UTF-8
Content-Length: 78
Content-Integrity:
  mac=cjkMkfnnYP8JYWZAbRLvtpqImmOK3rsrOT1XcvAgHDk=;
  ticket=TUMnorO0SjHHS7D2uFcGlRYJ0Hd3eibwe0ogptoNMQuCYmCHfHAJcJlyvi
    j8WoXDglTSOkctnmoBzl8W0NLSlcgSyZcmsAyoWs8y1Rn2ZlO2WBgoWrFIOqPa4
    oB29dgs/ei6ieINZtmvXNCm2NUkWA==

{
  "TicketRequest": {
    "ChallengeResponse": "TctLOG74cwpm26YNpEibcQ=="}}
```

If the response to the PIN challenge is correct, the service responds with a message that specifies a set of cryptographic parameters to be used to authenticate future interactions with the service (Cryptographic) and a set of connection parameters for servers supporting the Query Service (Service).

In this case the server returns three connections, each offering a different transport protocol option. Each connection specifies its own set of cryptographic parameters (or will when the code is written for that).

```
HTTP/1.1 200 Complete
Content-Type: application/json;charset=UTF-8
Content-Length: 1907
Content-Integrity:
  mac=nKhjR1r2eYPga0rmDfHT4HOvgQ+EuUoQPwzIl0btljs=;
  ticket=TUMnorO0SjHHS7D2uFcGlRYJ0Hd3eibwe0ogptoNMQuCYmCHfHAJcJlyvi
    j8WoXDglTSOkctnmoBzl8W0NLSlcgSyZcmsAyoWs8y1Rn2ZlO2WBgoWrFIOqPa4
    oB29dgs/ei6ieINZtmvXNCm2NUkWA==

{
  "TicketResponse": {
    "Status": 200,
    "StatusDescription": "Complete",
    "Cryptographic": [{
        "Protocol": "OBPConnection",
        "Secret": "HQuQg4GkzTwTVoGxar0EXg==",
        "Encryption": "A128CBC",
        "Authentication": "HS256",
        "Ticket":
        "0ulMVMMfY/pLHZ0FlIy2zDnNycYz9Znvs3JJYQGlZ+dWaxMNxm/jLEsJd/
        0qsAc5qp8fjBoMN49V9DkDgM4UYJxVriqfr64RyTTgug2taHY="}],
    "Service": [{
        "Name": "obp1.example.com",
        "Port": 443,
        "Address": "10.1.2.3",
        "Priority": 1,
        "Weight": 100,
        "Transport": "WebService",
        "Cryptographic": {
          "Protocol": "OBPQuery",
          "Secret": "kezeXxhkzXgxY7vpkHUb1g==",
          "Encryption": "A128CBC",
          "Authentication": "HS256",
          "Ticket":
          "jpBXvI7/0WTmwI2NN4n7Vvw96nbS9LpSsSNMIkdapiUoLikSkjpgMrtb
          VKz5lHOPloCgAyZXxfZpQRsp4oPY4BcRaMw6F5na62IHnBVDeXg="}},
      {
        "Name": "dns1.example.com",
        "Port": 53,
        "Address": "10.1.2.2",
        "Priority": 1,
        "Weight": 100,
        "Transport": "DNS",
        "Cryptographic": {
          "Protocol": "OBPQuery",
          "Secret": "Wk3m7DlL/GStBBm3xUjyzg==",
          "Encryption": "A128CBC",
          "Authentication": "HS256",
          "Ticket":
          "Q9r4hXefHhLvgpKHVg3w2p7VptVH9qidGiIa4Nw3Zp5hZR816h9+PRj5
          sye1jmIhy4sYA/jfK/g4OrSngK9xWO7Qg3/iQ+YTAchKQjdJtN4="}},
      {
        "Name": "udp.example.com",
        "Port": 5000,
        "Address": "10.1.2.2",
        "Priority": 1,
        "Weight": 100,
        "Transport": "UDP",
        "Cryptographic": {
          "Protocol": "OBPQuery",
          "Secret": "wBiguG9FGj08nS/c/njp4Q==",
          "Encryption": "A128CBC",
          "Authentication": "HS256",
          "Ticket":
          "F8LPKTL+XaAX0eJsM22fdJ37BRS816dKXD66UbD8NAVKOgOu556uS8WW
          AMj+dJbJaErUzo/vw7tY0icCu1bw8qHmOO4gzhbSbD4Nga2EAU4="}}]}
          }
```

When Alice's machine is to be transfered to another employee, the Unbind transaction is used. The only parameter is the Ticket identifying the device association (Ticket).

```
Post / HTTP/1.1
Host: example.com
Cache-Control: no-store
Content-Type: Application/json;charset=UTF-8
Content-Length: 25
Content-Integrity:
  mac=bZU61eCOW4nVnfdJNS719HL4IsNVxtoTgoRt+mqLbWY=;
  ticket=0ulMVMMfY/pLHZ0FlIy2zDnNycYz9Znvs3JJYQGlZ+dWaxMNxm/jLEsJd/
    0qsAc5qp8fjBoMN49V9DkDgM4UYJxVriqfr64RyTTgug2taHY=

{
  "UnbindRequest": {}}
```

Since the unbind response represents the termination the relationship with the Omnibroker, the response merely reports the success or failure of the request.

```
HTTP/1.1 0
Content-Type: application/json;charset=UTF-8
Content-Length: 26
Content-Integrity:
  mac=9P1FmroeFU7y9qHgXdSFXH2qSImh0cQpaSgZrx5IswM=;
  ticket=0ulMVMMfY/pLHZ0FlIy2zDnNycYz9Znvs3JJYQGlZ+dWaxMNxm/jLEsJd/
    0qsAc5qp8fjBoMN49V9DkDgM4UYJxVriqfr64RyTTgug2taHY=

{
  "UnbindResponse": {}}
```

The 'Ticket' value presented in the foregoing examples is a sequence of binary data generated by the service is opaque to the client. Services MAY generate ticket values with a substructure that enable the service to avoid the need to maintain server side state.

In the foregoing example, the ticket structures generated by the service encode the cryptographic parameter data, the shared secret, account identifier and an authentication value. The initial ticket value generated additionally encodes the values of the client and service challeng values for use in calculating the necessary ChallengeResponse.

3.2. OBPConnection

TOC

3.2.1. Message: Message

TOC

3.2.2. Message: Response

TOC

Status : Integer [0..1]
    Application layer server status code
StatusDescription : String [0..1]
    Describes the status code (ignored by processors)

3.2.3. Message: ErrorResponse

An error response MAY be returned in response to any request.

Note that requests MAY be rejected by the code implementing the transport binding before application processing begins and so a server is not guaranteed to provide an error response message.

3.2.4. Message: Request

Ticket : Binary [1..1]
: Opaque ticket issued by the server that identifies the cryptographic parameters for encryption and authentication of the message payload.

3.2.5. Structure: Cryptographic

Parameters describing a cryptographic context.

Protocol : Label [0..1]
: OBP tickets MAY be restricted to use with either the management protocol (Management) or the query protocol (Query). If so a service would typically specify a ticket with a long expiry time or no expiry for use with the management protocol and a separate ticket for use with the query protocol.

Secret : Binary [1..1]
: Shared secret

Encryption : Label [1..1]
: Encryption Algorithm selected

Authentication : Label [1..1]
: Authentication Algorithm selected

Ticket : Binary [1..1]
: Opaque ticket issued by the server that identifies the cryptographic parameters for encryption and authentication of the message payload.

Expires : DateTime [0..1]
: Date and time at which the context will expire

3.2.6. Structure: ImageLink

Algorithm : Label [0..1]
: Image encoding algorithm (e.g. JPG, PNG)

Image : Binary [0..1]
: Image data as specified by algorithm

3.2.7. Structure: Connection

Contains information describing a network connection.

Name : Name [0..1]
: DNS Name. Since one of the functions of an OBP service is name resolution, a DNS name is only used to establish a connection if connection by means of the IP address fails.

Port : Integer [0..1]
: TCP or UDP port number.

Address : String [0..1]
: IPv4 (32 bit) or IPv6 (128 bit) service address

Priority : Integer [0..1]
: Service priority. Services with lower priority numbers SHOULD be attempted before those with higher numbers.

Weight : Integer [0..1]
: Weight to be used to select between services of equal priority.

Transport : Label [0..1]
: OBP Transport binding to be used valid values are HTTP, DNS and UDP.

Expires : DateTime [0..1]
: Date and time at which the specified connection context will expire.

TOC

3.2.8. Bind

Binding a device is a two step protocol that begins with the Start Query followed by a sequence of Ticket queries.

TOC

3.2.9. Message: BindRequest

The following parameters MAY occur in either a StartRequest or TicketRequest:

Encryption : Label [0..Many]
: Encryption Algorithm that the client accepts. A Client MAY offer multiple algorithms. If no algorithms are specified then support for the mandatory to implement algorithm is assumed. Otherwise mandatory to implement algorithms MUST be specified explicitly.

Authentication : Label [0..Many]
: Authentication Algorithm that the client accepts. If no algorithms are specified then support for the mandatory to implement algorithm is assumed. Otherwise mandatory to implement algorithms MUST be specified explicitly.

TOC

3.2.10. Message: BindResponse

The following parameters MAY occur in either a StartResponse or TicketResponse:

Cryptographic : Cryptographic [0..Many]
: Cryptographic Parameters.

Service : Connection [0..Many]
: A Connection describing an OBP service point

TOC

3.2.11. Message: OpenRequest

The OpenRequest Message is used to begin a device binding transaction. Depending on the authentication requirements of the service the transaction may be completed in a single query or require a further Ticket Query to complete.

If authentication is required, the mechanism to be used depends on the capabilities of the device, the requirements of the broker and the existing relationship between the user and the broker.

If the device supports some means of data entry, authentication MAY be achieved by entering a passcode previously delivered out of band into the device.

The OpenRequest specifies the properties of the service (Account, Domain) and Device (ID, URI, Name) that will remain constant throughout the period that the device binding is active and parameters to be used for the mutual authentication protocol.

Account : String [0..1]
: Account name of the user at the OBP service

Domain : Name [0..1]

Domain name of the OBP broker service

HavePasscode : Boolean [0..1] Default =False
: If 'true', the user has entered a passcode value for use with passcode authentication.

HaveDisplay : Boolean [0..1] Default =False
: Specifies if the device is capable of displaying information to the user or not.

Challenge : Binary [0..1]
: Client challenge value to be used in authentication challenge

DeviceID : URI [0..1]
: Device identifier unique for a particular instance of a device such as a MAC or EUI-64 address expressed as a URI

DeviceURI : URI [0..1]
: Device identifier specifying the type of device, e.g. an xPhone.

DeviceName : String [0..1]
: Descriptive name for the device that would distinguish it from other similar devices, e.g. 'Alice's xPhone".

TOC

3.2.12. Message: OpenResponse

An Open request MAY be accepted immediately or be held pending completion of an inband or out-of-band authentication process.

The OpenResponse returns a ticket and a set of cryptographic connection parameters in either case. If the

Challenge : Binary [0..1]
: Challenge value to be used by the client to respond to the server authentication challenge.

ChallengeResponse : Binary [0..1]
: Server response to authentication challenge by the client

VerificationImage : ImageLink [0..Many]
: Link to an image to be used in an image verification mechanism.

TOC

3.2.13. Message: TicketRequest

The TicketRequest message is used to (1) complete a binding request begun with an OpenRequest and (2) to refresh ticket or connection parameters as necessary.

ChallengeResponse : Binary [0..1]
: The response to a server authentication challenge.

TOC

3.2.14. Message: TicketResponse

The TicketResponse message returns cryptographic and/or connection context information to a client.

TOC

3.2.15. Unbind

Requests that a previous device association be deleted.

TOC

3.2.16. Message: UnbindRequest

Since the ticket identifies the binding to be deleted, the only thing that the unbind message need specify is that the device wishes to cancel the binding.

3.2.17. Message: UnbindResponse

Reports on the success of the unbinding operation.

If the server reports success, the client SHOULD delete the ticket and all information relating to the binding.

A service MAY continue to accept a ticket after an unbind request has been granted but MUST NOT accept such a ticket for a bind request.

4. Mutual Authentication

A Connection Service MAY require that a connection request be authenticated. Three authentication mechanisms are defined.

> PIN Code: The client and server demonstrate mutual knowledge of a PIN code previously exchanged out of band.
>
> Established Key: The client and server demonstrate knowledge of the private key associated with a credential previously established. This MAY be a public key or a symmetric key.
>
> Out of Band Confirmation: The request for access is forwarded to an out of band confirmation service.

4.1. PIN Authentication

[Motivation]

Although the PIN value is never exposed on the wire in any form, the protcol considers the PIN value to be a text encoded in UTF8 encoding.

[Considerations for PIN character set choice discussed in body of draft, servers MUST support numeric only, clients SHOULD support full Unicode]

The PIN Mechanism is a three step process:

> The client sends an OpenRequest message to the Service containing a challenge value CC.
>
> The service returns an OpenResponse message containing to the client a server challenge value SV and a server response value SR.
>
> The client sends a TicketRequest message to the service containing a client response value CR.

Since no prior authentication key has been the OpenRequest and OpenResponse messages are initially sent without authentication and authentication values established the Challenge-Response mechanism.

The Challenge values CC, and SC are cryptographic nonces. The nonces SHOULD be generated using an appropriate cryptographic random source. The nonces MUST be at least as long as 128 bits, MUST be at least the minimum key size of the authentication algorithm used and MUST NOT more than 640 bits in length (640 bits should be enough for anybody).

The server response and client response values are generated using an authentication algorithm selected by the server from the choices proposed by the client in the OpenRequest message.

The algorithn chosen may be a MAC algorithm or an encrypt-with-authentication (EWA) algorithm. If an EWA is specified, the encrypted data is discarded and only the authentication value is used in its place.

Let A(d,k) be the authentication value obtained by applying the authentication algorithm with key k to data d.

To create the Server Response value, the UTF8 encoding of the PIN value 'P' is first converted into a symmetric key KPC by using the Client challenge value as the key truncating if necessary and then applied to the of the OpenRequest message:

KPC = A (PIN, CC) SR = A (Secret + SC + OpenRequest, KPC)

In the Web Service Binding, the Payload of the message is the HTTP Body as presented on the wire. The Secret and Server Challenge are presented in their binary format and the '+' operator stands for simple concatenation of the binary sequences.

This protocol construction ensures that the party constructing SR:

> Knows the PIN code value (through the construction of KPC).
>
> Is responding to the Open Request Message (SR depends on OpenRequest).
>
> Has knowlege of the secret key which MUST be used to authenticate the following TicketRequest/TicketResponse interaction that will establish the actual connection.
>
> Does not provide an oracle the PIN value. That is, the protocol does not provide a service that reveals the (since the value SR includes the value SC which is a random nonce generated by the server and cannot be predicted by the client).

To create the Client Response value, secret key is applied to the PIN value and server Challenge:

CR = A (PIN + SC + OpenRequest, Secret)

Note that the server can calculate the value of the Client Response token at the time that it generates the Server Challenge. This minimizes the amount of state that needs to be carried from one request to the next in the Ticket value when using the stateless server implementation described in section **Section 4.4**

This protocol construction ensures that the of CR

> Knows the PIN value.
>
> Is respoding to the OpenResponse generated by the server.

Note that while disclosure of an oracle for the PIN value is a concern in the construction of CR, this is not the case in the construction of SR since the client has already demonstrated knowledge of the PIN value.

4.2. Example: Latin PIN Code

TOC

The Connection Request example of section **Section 3.1.3** demonstrates the use of an alphanumeric PIN code using the Latin alphabet.

The PIN code is [] and the authentication algorithm is []. The value KP is thus:

[TBS]

The data over which the hash value is calulated is Secret + SC + OpenRequest:

[TBS]

Applying the derrived key to the data produces the server response:

The data for the client response is PIN + SC:

[TBS]

Applying the secret key to the data produces the client response:

[TBS]

4.3. Example: Cyrillic PIN Code

TOC

If the PIN code in the earlier example was [] the value KP would be:

[TBS]

The Server Response would be:

[TBS]

The rest of the protocol would then continue as before.

4.4. Stateless server

TOC

The protocol is designed to permit but not require a stateless implementation by the server using the Ticket value generated by the server to pass state from the first server transaction to the second.

In the example shown in **Section 3.1.3**, the server generates a 'temporary ticket' containing the following information:

If a server uses the Ticket to transmit state in this way it MUST protect the confidentiality of the ticket using a strong means of encryption and authentication.

4.5. Established Key

TOC

The Established Key mechanism is used when the parties have an existing shared key or public key credential.

The [Open request open response are authenticated under the respective keys]

SR=CC, CR=SC

4.6. Out of Band Confirmation

TOC

The Out Of Band Confirmation mechanism is a three step process in which:

> The client makes an OpenRequest message to the service and obtains an OpenResponse message.
>
> The service is informed that the service has been authorized through an out of band process.
>
> The client makes a TicketRequest to the service and obtains a TicketResponse message to complete the exchange.

Since no prior authentication key has been the OpenRequest and OpenResponse messages are sent without authentication.

The principal concern in the Out Of Band Confirmation mechanism is ensuring that the party authorizing the request is able to identify which party originated the request they are attempting to identify.

If a device has the ability to display an image it MAY set the HasDisplay=true in the OpenRequest message. If the broker recieves an OpenRequest with the HasDisplay value set to true, the OpenResponse MAY contain one or more VerificationImage entries specifying image data that is to be displayed to the user by both the client and the confirmation interface.

Before confirming the request, the user SHOULD verify that the two images are the same and reject the request in the case that they are not.

Many devices do not have a display capability, in particular an embedded device such as a network switch or a thermostat. In this case the device MAY be identified by means of the information provided in DeviceID, DeviceURI, DeviceImage and DeviceName.

5. Acknowledgements

TOC

[List of contributors]

6. Security Considerations

TOC

6.1. Denial of Service

TOC

6.2. Breach of Trust

TOC

6.3. Coercion

TOC

7. To do

TOC

Formatting of the abstract data items needs to be improved

8. IANA Considerations

TOC

[TBS list out all the code points that require an IANA registration]

9. Normative References

TOC

[RFC1035] Mockapetris, P., "**Domain names - implementation and specification**," STD 13, RFC 1035, November 1987 (**TXT**).

[RFC2119] **Bradner, S.**, "**Key words for use in RFCs to Indicate Requirement Levels**," BCP 14, RFC 2119, March 1997 (**TXT**, **HTML**, **XML**).

[RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "**Transport Layer Security (TLS) Extensions**," RFC 4366, April 2006 (**TXT**).

[X.509] International Telecommunication Union, "**ITU-T Recommendation X.509 (11/2008): Information technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks**," ITU-T Recommendation X.509, November 2008.

[X.680] International Telecommunication Union, "**ITU-T Recommendation X.680 (11/2008): Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation**," ITU-T Recommendation X.680, November 2008.

Appendix A. Example Data.

TOC

A.1. Ticket A

TOC

A.2. Ticket B

TOC

Author's Address

TOC

Phillip Hallam-Baker
Comodo Group Inc.
Email: phillip@comodo.com