Internet Engineering Task Force (IETF)
Internet-Draft
Intended Status: Standards Track
Expires: November 22, 2014

Phillip Hallam-Baker
Comodo Group Inc.

May 21, 2014

OmniBroker Publication Protocol
draft-hallambaker-omnipublish-00

Abstract

OmniPublish is a Web Service that supports server configuration management. The supported transaction set allows a server to obtain and renew necessary cryptographic credentials, publish service discovery statements and obtain network configuration specifications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

Copyright Notice

Table of Contents

1. Definitions

1.1. Requirements Language

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 [RFC2119].

2. Introduction

OmniPublish is a Web Service that supports server configuration management. The supported transaction set allows a server to obtain and renew necessary cryptographic credentials, publish service discovery statements and obtain network configuration specifications.

The services supported by OmniPublish are complimentary to the services provided by OmniBroker [I-D.hallambaker-omnibroker] and the protocols share the same transport binding (HTTP, UYFM) and encoding options (JSON).

2.1. Traditional Server Configuration Approach

In the traditional approach to server configuration the network administrator is required to anticipate and perform all the necessary configuration needs of the service. For an enterprise server these steps will typically include:

* Enter server parameters in the DNS.
* Configure firewall to permit external access.
* Generate Public/Private Keypair.
* Apply for digital certificate.
* Install digital certificate.

While executing each individual step may be considered straightforward, any configuration task involving five non-trivial human mediated tasks is liable to be error prone. Moreover maintaining the configuration represents an ongoing maintenance effort as certificates expire, network configurations are changed, servers are updated, etc.

2.2. Automating network management.

In the traditional administration model the human is required to anticipate the needs of the server. Yet the server itself knows its needs with great precision although not necessarily how they are to be realized.

A server that is configured to use the TLS protocol knows that a certificate is required and the purposes for which it is to be used. It knows when the certificate is about to expire and requires replacement and when evidence of certificate status (e.g. an OCSP token) requires renewal.

Network configuration raises similar considerations except that the information available to a server is typically insufficient to perform network configuration tasks. It is not guaranteed that the local network IP address of a server is the same as the IP address that is visible to the external network. A mechanism in which the server edits DNS entries directly is therefore less functional than one in which the DNS entries are generated by a mediated service that has access to the necessary additional data.

Network configuration is an administration function and therefore requires administrative privileges. Accordingly, every OmniPublish request and response is authenticated using credentials established using the SXS-Connect protocol [I-D.hallambaker-wsconnect].

2.2.1. Cloud Computing Requirements.

Cloud computing does not necessarily raise new management requirements but the requirements that are raised become more urgent. In the traditional model a service ran on a fixed number of hosts in a configuration that was static for months or years. In a cloud computing environment the number of hosts supporting a service might vary several times in an hour to respond to variations in load.

An important consequence of the transient nature of cloud computing is that hosts which provide a service for a few hours or even minutes are issued cryptographic credentials that are valid for a year.

3. Omnibroker Publication Service

The OmniPublish service is designed to permit services to manage themselves to the greatest extent possible.

The features that are most likely to make deployment attractive in the short term are the ability to manage cryptographic credentials including acquisition of public/private keypairs, certificates and certificate status assertions.

An enterprise with a large in-house IT department would typically host the Omnipublish service locally. The local service would then be configured to forward publication data to any IT facitilities that happen to be outsourced such as CA services, DNS etc.

A similar model may be applied in the home automation environment with devices under management publishing their service information to

the local publication server which then forwards the information to

external services as necessary. The chief difference between this case and the enterprise case being that the service operation cannot depend on the end user being aware that the device exists, let alone perform configuration.

In a pure cloud computing environment the OmniPublish service would have to be outsourced since there is no internal IT system for it to run off.

3.1. Service Binding

Application establishes a service connection to the OmniPublish service.

3.2. Acquiring Cryptographic Credentials

One of the chief reasons given for not using cryptographic protocols such as IPSEC, S/MIME and TLS is the difficulty of obtaining or registering the necessary cryptographic credentials. In the case of an Internet protocol this is typically but not always a PKIX certificate bound to a private key held by the the certificate subject.

3.2.1. Example: Small Web Site Operator

Alice is the owner of a small business that operates a Web site. To protect the privacy of the Web site users, Alice decides to enable TLS on the Web site. Accordingly, Alice selects a Certificate Authority Example CA Inc. that issues a certificate with an appropriate validation requirement for her intended use.

Alice provides her contact details to the CA which returns an account identifier alice@example.net and a PIN value [TBS].

The credentials are immediately valid for creating a service connection using the PIN. The Service Connection Ticket is obtained by a Web Server administration tool:

```
POST /.well-known/sxs-connect/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Host: localhost:8080
Content-Length: 344
Expect: 100-continue

{
  "OpenPINRequest": {
    "Encryption": ["A128CBC",
      "A256CBC",
      "A128GCM",
```

```
          "A256GCM"],
```

```
      "Authentication": ["HS256",
        "HS384",
        "HS512",
        "HS256T128"],
      "Account": "alice",
      "Service": ["omni-publish"],
      "Domain": "example.net",
      "HaveDisplay": false,
      "Challenge": "
  4m7Lzr7g2FzllXcVGeDIOw"}}
```

The service responds with the challenge to be used to validate the PIN:

```
  HTTP/1.1 281 Pin code required
  Content-Length: 511
  Date: Wed, 21 May 2014 20:05:54 GMT
  Server: Microsoft-HTTPAPI/2.0

  {
    "OpenPINResponse": {
      "Status": 281,
      "StatusDescription": "Pin code required",
      "Challenge": "
  cHbxV3Uwkb-CYezJhKj-wA",
      "ChallengeResponse": "
  98RV4Se7VQIP3FbqcrLKyUth5u6F48dbCGpzrHpkGfQ",
      "Cryptographic": {
        "Secret": "
  e6oSWl3dFfNkpYXvSTvY1w",
        "Encryption": "A128CBC",
        "Authentication": "HS256",
        "Ticket": "
  V8ae-8uQMtt_uyKJQLbx4umJEpsz--OXVriEHRoq5sw5uq6u1_4tWv8ro7DyD5Su
  hSpOibX2cOnd0OHSOJpcA1Gs9WjRArqzz0WrD0Inl39d89zbcWoMKYKhlOFFV_LF
  V8kPPoK8BmaQOxCo3kBrxg"}}}
```

The administration tool completes the request by proving knowledge of the PIN:

```
  POST /.well-known/sxs-connect/ HTTP/1.1
  Content-Type: application/json;charset=UTF-8
  Cache-Control: no-store
  Session: Value=J5wXEchUbr7k2A0mvaOEPnr3KGFagzg1vH_MX6W1R14;
    Id=V8ae-8uQMtt_uyKJQLbx4umJEpsz--OXVriEHRoq5sw5uq6u1_4tWv8ro7Dy
    D5SuhSpOibX2cOnd0OHSOJpcA1Gs9WjRArqzz0WrD0Inl39d89zbcWoMKYKhlOF
    FV_LFV8kPPoK8BmaQOxCo3kBrxg
  Host: localhost:8080
  Content-Length: 129
```

```
   Expect: 100-continue
```

```
{
  "TicketRequest": {
    "Service": ["omni-publish"],
    "ChallengeResponse": "
49GCx5HUvU2SNE6M3GuKcxgFfvZKDLpTfpqXUOAGXVE"}}
```

The Connection service returns the OmniPublish connection parameters:

```
HTTP/1.1 OK Success
Content-Length: 1306
Date: Wed, 21 May 2014 20:05:54 GMT
Server: Microsoft-HTTPAPI/2.0

{
  "TicketResponse": {
    "Status": 200,
    "StatusDescription": "Success",
    "Cryptographic": [{
        "Protocol": "sxs-connect",
        "Secret": "
DOdZw6ynGANAjqnR-1gL0A",
        "Encryption": "A128CBC",
        "Authentication": "HS256",
        "Ticket": "
Ay9sUcNcC0GH9cY5NiRFcrqjFL0wTgTn69uO9SUPvP_XqB05yJ_fLqeI622H-bBu
klDhb1TpUGwQMAgNNwRkgsu97Dc38WBfUiyetM0TwYY"}],
    "Service": [{
        "Service": "omni-publish",
        "Name": "localhost",
        "Port": 8080,
        "Priority": 100,
        "Weight": 100,
        "Transport": "HTTP",
        "Cryptographic": {
          "Secret": "
LfRHVDFWVkVQu81lI2wT8w",
          "Encryption": "A128CBC",
          "Authentication": "HS256T128",
          "Ticket": "
OufJWkZCHmCLVeSuS4Nth4ozUrRfyDa4v8Dd5FIrkIWPQrGnHLgG_6ZmoHQGqIRg
7BL7Gm9Jq7LQihkCLhgQjp1LJqpDmDuDFbH5ZRDl7_g"}},
      {
        "Service": "omni-publish",
        "Name": "localhost",
        "Port": 9090,
        "Priority": 100,
        "Weight": 100,
        "Transport": "UDP",
        "Cryptographic": {
```

```
            "Secret": "
   elODZJePf2UDWW1hHw2-3A",
```

```
            "Encryption": "A128CBC",
            "Authentication": "HS256T128",
            "Ticket": "
   xMn4JDCQIg9WSHTVh1HwdJQq1iBoIHNk-BRHdhq_WGGeWv6cgfDgLYo2-U4BX2IH
   _SRzZ1fDf5dHzfE67wuPawutwOkemJNH4mOK0XYeNPc"}}]}}
```

The administration tool enters the connection parameters into the server configuration data. At this point all the administrative tasks related to the server are complete and the remainder of the process can be performed automatically.

The server begins the process by generating a public key pair and Certificate Signing Request [RFC2986] and requests issue of a certificate with a CredentialRequest:

```
POST /.well-known/omni-publish/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=ArwY9yiAOaMjxTx4jE_BzNTNJ5z4Nn-I6gjdZPC5ejI;
  Id=OufJWkZCHmCLVeSuS4Nth4ozUrRfyDa4v8Dd5FIrkIWPQrGnHLgG_6ZmoHQG
  qIRg7BL7Gm9Jq7LQihkCLhgQjp1LJqpDmDuDFbH5ZRDl7_g
Host: localhost:8080
Content-Length: 148
Expect: 100-continue

{
  "CredentialRequest": {
    "Authentication": {
      "ContentType": "application/pkcs-10",
      "Data": "
AQID"},
    "MakePrivateKey": false}}
```

The service accepts the request but the process cannot be completed until the validation process required for the class of certificate has been completed. Accordingly the service returns the status 'Pending' and gives an estimated completion time:

```
HTTP/1.1 282 Transaction Incomplete
Content-Length: 98
Date: Wed, 21 May 2014 20:05:55 GMT
Server: Microsoft-HTTPAPI/2.0

{
  "CredentialResponse": {
    "Status": 282,
    "StatusDescription": "Transaction Incomplete"}}
```

The validation process completes successfully and the CA issues the

certificate. The server requests delivery of the certificate by repeating the CredentialRequest:

```
POST /.well-known/omni-publish/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=ArwY9yiAOaMjxTx4jE_BzNTNJ5z4Nn-I6gjdZPC5ejI;
  Id=OufJWkZCHmCLVeSuS4Nth4ozUrRfyDa4v8Dd5FIrkIWPQrGnHLgG_6ZmoHQG
  qIRg7BL7Gm9Jq7LQihkCLhgQjp1LJqpDmDuDFbH5ZRDl7_g
Host: localhost:8080
Content-Length: 148
Expect: 100-continue

{
  "CredentialRequest": {
    "Authentication": {
      "ContentType": "application/pkcs-10",
      "Data": "
AQID"},
    "MakePrivateKey": false}}
```

This time the certificate is ready and is returned to the server. For the convenience of the server software, the response message tells the Web server when the certificate will expire and the earliest and latest dates on which to request renewal:

```
HTTP/1.1 282 Transaction Incomplete
Content-Length: 98
Date: Wed, 21 May 2014 20:05:55 GMT
Server: Microsoft-HTTPAPI/2.0

{
  "CredentialResponse": {
    "Status": 282,
    "StatusDescription": "Transaction Incomplete"}}
```

Note that the certificate returned is a short lifetime certificate that is only valid for a 72 hour interval, 24 hours of which have already elapsed at issue time. Use of short lived certificates is generally accepted as being highly desirable as it eliminates the need for certificate status reporting. The certificates issued will expire at the same time that any static status report would. The chief objection to the use of short lived certificates has been the need for daily administrative intervention. Automating the process of updating the certificate eliminates this objection.

In addition to eliminating the need to track revocation status separately, performing certificate updates on a daily basis is potentially more reliable than one that is only activated once a year. Network changes that prevent a an update completing successfully have immediate impact at a time the network

administration is looking for potential problems rather than being discovered up to a year later when the personel who caused the change

may have been reassigned or left the company.

The server MAY apply for renewal of the certificate at any time after the earliest date specified in the issue statement. If no request is made by the time that the latest time has been reached, the issuing CA MAY begin attempting to contact their customer to determine the cause. To avoid unnecessary warning messages from the CA (and possibly additional invoices for unused services) the server may inform the CA that certificate updates will not be required for an extended period using the Notify method:

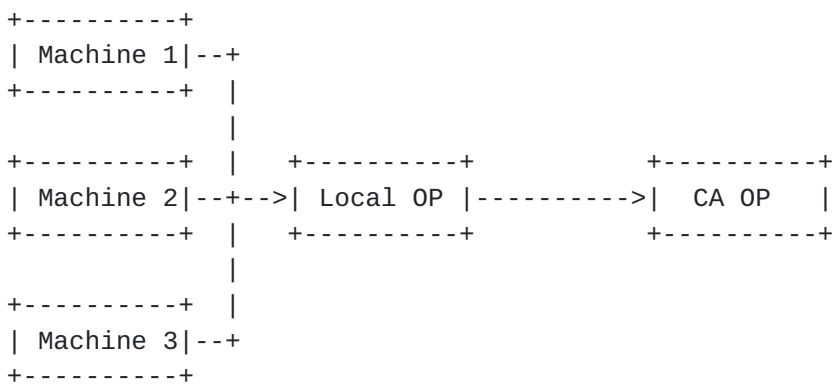
```
POST /.well-known/omni-publish/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=wFsqI6wkH-TuCyGkIOjL3TJsbkvJCXxdHGohugk0hx0;
  Id=vVuUnw2Pi0xlHB2OFnbOBTDmLn9qC1HhEjUkJMHfCtBZRCPXc1GzPw8TLm1b
  8asS-GCtD8H681WvoGW0DcEgNMDUc0UZu-ZPo_wA9f8f-bk
Host: localhost:8080
Content-Length: 129
Expect: 100-continue

{
  "NotifyRequest": {
    "NextState": "Offline",
    "Earliest": "2014-05-21T20:05:56Z",
    "Latest": "2014-05-21T20:06:56Z"}}
```

3.2.2. Example: Large Enterprise

Since Alice only operates one Web server, the simplest management solution for her is for the Web server to establish a direct connection to the CA. In a large enterprise with several hundred servers, a centralized management approach which allows configurations to be applied to groups of servers as a unit is usually required.

To support this configuration, Bob deploys a local OmniPublish service in his network. Every machine that Bob manages connects to his local OmniPublish service to obtain its cryptographic credentials. The local OmniPublish service connects to the OmniPublish service of the CA to these service requests:



3.3. Generating or Obtaining a Public/Private KeyPair.

Conventional wisdom holds that public/private key pairs should be generated on the host on which they are to be used and exist in no other location.

In practice, this mode of operation is not always the most desirable. In the case of keypairs to be used for encryption of static data, the decryption key must be available to all the machines that need decryption capabilities.

Key generation procedures for public key algorithms can be lengthy. While a delay of a few seconds or even a few minutes is acceptable in a one-time server configuration process, introducing such a delay into server startup is frequently unacceptable.

Experience of operating cryptographic systems has proved that correct and secure implementation of key generation capabilities is beyond the capabilities of many programmers. Random seeds are frequently generated with insufficient entropy. In some cases entropy is leaked after the seed is used to generate the private key.

For the above reasons, it is frequently but not always desirable to perform generation of public/private keypairs as a centralized service supported by a small number of machines that can be tightly controlled an audited.

3.3.1. Example: Internet Coffee Pot

Bob buys a new coffee pot for his office that supports the hypothetical 'Ready to Brew' Web Service that allows the machine to be instructed to brew a cup of fresh coffee. Since this is an important and security sensitive function, the coffee pot supports use of the TLS protocol but the control hardware does not have access to a suitible source of randomness for generating a public keypair.

To meet this need the coffee pot simply requests that the OmniPublish service generate a keypair on its behalf and return the private key

with the certificate. Note that since Bob has bound the coffee pot to

his local omnibroker service rather than a service provided by a public CA, Bob still exercises full control over generation of public/private keypairs. He is simply choosing to generate the keypair in a different place.

[TBS: provide a DH exchange to enable an application level guarantee that the private key is delivered under a sound encryption scheme]

```
POST /.well-known/omni-publish/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=tKs0Y7AGoqfFtZDIUh395TlbIU2E6ciO2beA4lnBtgs;
  Id=vVuUnw2Pi0xlHB2OFnbOBTDmLn9qC1HhEjUkJMHfCtBZRCPXc1GzPw8TLm1b
  8asS-GCtD8H681WvoGW0DcEgNMDUc0UZu-ZPo_wA9f8f-bk
Host: localhost:8080
Content-Length: 147
Expect: 100-continue

{
  "CredentialRequest": {
    "Authentication": {
      "ContentType": "application/pkcs-10",
      "Data": "
AQID"},
    "MakePrivateKey": true}}
```

The service accepts the request and returns the requested credentials:

```
HTTP/1.1 OK Success
Content-Length: 3426
Date: Wed, 21 May 2014 20:05:55 GMT
Server: Microsoft-HTTPAPI/2.0

{
  "CredentialResponse": {
    "Status": 200,
    "StatusDescription": "Success",
    "Credential": {
      "ContentType": "application/pkix-cert",
      "Data": "
eA0QIvK3hMNUo3d8IxEchavHUltR5O1LUmEnLCcpGL3XROcD_3GtNpvEWgexojbi
By4SynEEzCGpRzyFoioPB1Fk2yrA_c74YxRYc4OuFryF9CgrbshEMHi-I9Szip1i
Lr6_NDwifiMAUH4KZEwj6TyVCh0zMHWtYY6T_itwdbZhOdxsSxITn4xBEUEzQs3w
mSJ5pRbhguaotJ_Vexv87eYlmn-nKrh9w99hVsNFWm2pVmWMgMF__Q_xGWpf4y_Z
S4Bko6jpHqY6MA_JjSBxrKXP2FG1JNCe-10I1Oohdtt7z3i1pi9nYb0opPL1pmj2
6uXHRYO8hy0oewXEdHP_zg"},
    "Support": [{
        "ContentType": "application/pkix-cert",
```

```
          "Data": "
   4IvLI64hE-6_UxpMD_GUkyJLjGlq3Hz9i7G8kOJkn6kSjqyWmy_MGLN57dDGnK7D
```

z9DJYjSHbPk6SaX3r6_ye5YSxyuVF2duQACtH4Bd3icq_QpfiuXB-l8c5QTJe60u
ZZgVsGhzpagNpIUEb4VlPVIuQ4ZdV-yxLWhYM28ibzhHMfxNo6YOw-Xa7ySujry4
kr-ojsARBcS6jys6-k0_KUH8WPkIeiBisNQI7c4IwhgCE1DJqZRfIEB4fTjLWV3-
frOuuY6Y1cz_whODCgn68phD2D6eFuPyiJncU6WrFxF_aQibTQ9C7C61SWtloM5r
ASUBjY-bD9_iCppEtHxzoA"},
{
"ContentType": "application/pkix-cert",
"Data": "
LP9gFTTrsaeA_y9GqH_3eNiIbVR6H3HPhEIgu0UlWIKQoy_PeQB3JsOCd5O2Ra2S
HnPBMG4aRCVBU8MOTk27L9t4jvHbIy0kC0Ja4hm5yedkcG8sPhFZkHIOmhLhuaNT
bdLWFIUkvus420fl6gw1DU9n1H9vCbxG21SXet7iEhXQ9e8tA_i_9046NuS1CwTG
kpX0JJqrHHiZ6GQ3kKcn5PAdZkWiMAHReSATeD8GblAwXCkvguvVodMRV84n7gp4
JMuUnpKbtpZ_x0ycbG6LhwjmXyT_GzAYtjmSFXsS1shzNkmTYLbus1QFDVWaVPFk
AFwfthWawhaYBj0JWiAA0A"},
{
"ContentType": "application/pkix-cert",
"Data": "
DInUnJXe53hXiwrrWEyZPOlKsjzEYnhN_eVIGkArTjR9L7WQ32jY1Mt6sfnQsYiM
cRfqANQd3tVLRKnyNQYJaU1MkNmOC7OLSLbWW4XBFxQaCk__T0IoDMxTP4Ol0uUW
EVo47OkUMnteOQ21qhBBW415fB2S_WN-UdB3ILzW8jfdNPVy1ctn5XjcaV_WfTPY
goUCtuJbULrB17LzmAYkNTdIny2uQs6vO8kogQH1jTw0WYb24NB_iO0tIw8gwRXb
iVb3cKH46ywNq4B2BjVcQw5UE_-Q-FMfArKTGg7Z0Oobg3VeOuX_3tC1_wRmnZjy
lGLHTGi5yU5nOVranbCuIQ"},
{
"ContentType": "application/ocsp-response",
"Data": "
iyqwOV3TFQE7e0KIPgtDZ0_rMPZSB0rnSBjII_07JwuJjPYpBv6R8U6uK4vJwHOy
mxCBQLwa8ZQf9wLUAXwaz1H0bN7vKRZgIoZsTrT96KHRVj1i867zS70ZZg0nH9X9
1dwuv_n7KUEmwoUqFX4x9B2Ug3z1TPv-iLjkpPyWz4g"},
{
"ContentType": "application/ocsp-response",
"Data": "
BYAKOKAi9fD3tni3pUKZxHfiBv0ICo9ULKBMiwPOGpGRVomwuawD-D-P4uDnEvXS
3mSa3fQ_fz2tw7fOCMrG3n16Yj0NCc4X8GHn49hRRTfzywsgybTxMitgSrXGh4us
Qao5gsnlR-Zo4oT2I1wWP1P9CrJY3KcGrSPpu40HD-4"},
{
"ContentType": "application/ocsp-response",
"Data": "
MqSNwx7iFJomrYmB6bo600I5rbEchtbQrkzyBZt7ZW79RI1NRZMN9tkYif4b_Gby
QNLNk3eY6WhooU7Yl9boIoQYas-SY7s8Njp4gQmIjk2nPNKNNn2qtrVbfYRtxEIg
Nm3zJy9CtdyU87zQxXG-oG29FBG-hAQjmNtEes4xgtU"},
{
"ContentType": "application/ocsp-response",
"Data": "
J5vbsOlkSk4S7CasMzjQOW5C4QXJXjztHP_MaGMvWF0C4CQxZkn_6lmIENuro2gv
ew3LTTvAv4ljESkcBP1iT7fGcQBE621ZO_8_RLn8XtDFmUyWvayguhvuhp4mwuL4
fFYXvwdWWVt_15X1HL1uCaOyXA88SQpjpxGN8ZuAtBc"},
{
"ContentType": "application/pkcs-12",

```
          "Data": "
   PUu_AzFikCBEYq1jB4d5pxCEsnvhq9iW4sXwf--3E8n-qr7HPEWXU2HWqtTjXA0E
```

1NQDWhJrxFM-o-EK1_gVGXKuSaGjndRVUm1p0ec-Vb5C5EBD4a0509ky0JaT1iMT
O-sCgBWkfHd-SOjSCbZxpTVZX_na3M0D12uoKCRDJcc39bDWebKrw3IEoJPFbuEn
PzGMmoYcs6cVpSl18BCj6t6-rZTSyIVk11NBuhZZ8CotbmdcqKs4_ORaufBgzNzd
o9rE-84MCBh-H3IkNZTuuakWOngdgSPkiOUuDd7k7YK-JNXwpDFWJeTiNxJiLOKE
vzPo8alcstgdnnb9dVu6uIz_-SXNMj7L5N3iuNRx8ZgWIaCVRDes6HuLqnEZXL-a
B1ZxUUvTPs-DUoNndpjsy5k5T1lwLuw7zCevqEBkxb1WpX4T8QBlDpLq1xjxW7tC
LUJyrCTRNrAP2t0OMo8z32_V3UZ5Qd5dwTELCfMqUAbDj9dZsKK4mhcdE6hArkr0
_XkzrIcL"}]}}

3.4. Request Network Configuration

Configuring a network server typically requires an administrator to perform several tasks:

* Assign an IP address to the server.
* Configure the firewall to accept incomming traffic and direct it to the server.
* Assign a DNS address to the server.
* Configure the server to accept connections at the chosen DNS address.
* Configure the relevant authoritative DNS server to publish the server records for the chosen DNS address.
* Update local LDAP directories.

While each individual step is straightforward, any error or inconsistency introduced may cause the configuration to fail or worse, succeed unreliably. Diagnosing and correcting such errors is one of the principal challenges in network administration.

Delegating responsibility for the network configuration to a service enables the configuration to be performed automatically and separates the configuration of the network from the configuration of the servers and services that use the network. This approach greatly simplifies deployment of complex network changes and makes major changes possible without interruption of service.

3.4.1. Example: Coffe Pot Service Registration.

Having deployed an infrastructure to automate management of his PKI credentials, Bob can leverage the same infrastructure to automate network configuration tasks as well.

The coffee pot establishes a service connection using the out of band authentication technique described in [I-D.hallambaker-wsconnect]. Having established the service connection, the coffee pot requests

advertisement of the brew-coffee Web service as follows:

```
POST /.well-known/omni-publish/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=cWUCjw6DTbyS8o1jkKSfLsJWb_8icI_HTb1Tpb80FJo;
  Id=vVuUnw2Pi0xlHB2OFnbOBTDmLn9qC1HhEjUkJMHfCtBZRCPXc1GzPw8TLm1b
  8asS-GCtD8H681WvoGW0DcEgNMDUc0UZu-ZPo_wA9f8f-bk
Host: localhost:8080
Content-Length: 313
Expect: 100-continue

{
  "AdvertiseRequest": {
    "Service": [{
        "Identifier": [{
            "Name": "Example.com",
            "Service": "_make_coffee._wks."}],
        "Connection": {
          "IPAddress": "10.1.2.3",
          "IPPort": 666,
          "Transport": "TLS",
          "TransportPolicy": "TLS=Required"}}]}}
```

The advertisement request succeeds and the OmniPublish service reports the successful outcome:

```
POST /.well-known/omni-publish/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=cWUCjw6DTbyS8o1jkKSfLsJWb_8icI_HTb1Tpb80FJo;
  Id=vVuUnw2Pi0xlHB2OFnbOBTDmLn9qC1HhEjUkJMHfCtBZRCPXc1GzPw8TLm1b
  8asS-GCtD8H681WvoGW0DcEgNMDUc0UZu-ZPo_wA9f8f-bk
Host: localhost:8080
Content-Length: 313
Expect: 100-continue

{
  "AdvertiseRequest": {
    "Service": [{
        "Identifier": [{
            "Name": "Example.com",
            "Service": "_make_coffee._wks."}],
        "Connection": {
          "IPAddress": "10.1.2.3",
          "IPPort": 666,
          "Transport": "TLS",
          "TransportPolicy": "TLS=Required"}}]}}
```

In this instance the OmniPublish service has granted the coffee pot a

48 hour lease on the service advertisement which must be renewed before expiry. In this case the publication request requires updates

to the DNS service which will take some time to propagate. An estimate of the time required to complete publication is returned.

4. OBPPublish

The OmniPublish protocol is a Web service that a network service or peer calls as a client to advertise the availability of a service and to obtain necessary cryptographic credentials.

4.1. OBPPublish Transactions

4.1.1. Advertise

* Request: AdvertiseRequest

* Response: AdvertiseResponse

Advises a broker that one or more Internet services are being offered with particular attributes.

4.1.2. Credential

* Request: CredentialRequest

* Response: CredentialResponse

Request issue of a cryptographic credential and (optionally) generate a public keypair

4.1.3. Notify

* Request: NotifyRequest

* Response: NotifyResponse

Notify the publication service that a server state transition has occurred or is planned.

4.2. OBPPublish Messages

4.2.1. Message: AdvertiseRequest

Specifies the connection(s) to be established.

The attributes required depend on the infrastructure(s) that the broker is capable of registering the service with.

Service :
OBPQuery.Service [0..Many] Describes a connection to be established.

4.2.2. Message: AdvertiseResponse

Specifies the connection(s)

Status :
Integer [1..1] Status return code value

StatusDescription :
String [0..1] Describes the status code (ignored by processors)

Service :
OBPQuery.Service [0..Many] Describes a connection that was established.

4.2.3. Message: CredentialRequest

Request issue of a cryptographic credential and (optionally) generate a public keypair.

SubjectIdentifier :
String [0..1] The DNS domain or [!RFC2822] account for which the credential is requested.

Authentication :
TaggedBinary [0..1] Data required by the credential issuer to authenticate the request. For example a Certificate Signing Request [!RFC2986].

MakePrivateKey :
Boolean [0..1] If true, requests that a private keypair be generated and the private component returned to the requestor.

ResponseTypes :
String [0..Many] Types of data requested in response.

4.2.4. Message: CredentialResponse

Returns issued cryptographic credentials.

Status :
Integer [1..1] Status return code value

StatusDescription :
String [0..1] Describes the status code (ignored by processors)

Credential :
TaggedBinary [0..1] The requested credential type, typically a PKIX End Entity certificate.

Support :
TaggedBinary [0..Many] Supporting data for the issued credential. For example one or more chains of certificate signing certificates, OCSP [!RFC6960] tokens etc.

SecretKey :
TaggedBinary [0..1] The secret key for the requested credential (if requested).

Expires :
DateTime [0..1] The time at which the credential will cease to be accepted by relying parties.

EarliestRenewal :
DateTime [0..1] The earliest time at which the issuer will accept renewal.

LatestRenewal :
DateTime [0..1] The latest time at which the issuer suggests requesting renewal.

4.2.5. Message: NotifyRequest

CurrentState :
String [0..1] Current state of the requestor

NextState :
String [0..1] State that the Requestor plans to enter

Earliest :
DateTime [0..1] Earliest time at which the transition is expected to complete

Latest :
DateTime [0..1] Latest time at which the transition is expected to complete

4.2.6. Message: NotifyResponse

Status :
Integer [1..1] Status return code value

StatusDescription :
String [0..1] Describes the status code (ignored by processors)

4.3. OBPPublish Structures

4.3.1. Structure: TaggedBinary

A sequence of values of the same type.

ContentType :
String [0..1] MIME Content Type of Data

Data :
Binary [0..1] Opaque binary data

5. Transport Bindings and Identifiers

The transport binding options for Omnibroker Publication are identical to those offered for Omnibroker Discovery [I-D.hallambaker-omnibroker].

5.1. Content-Type Identifiers

The following content identifiers are defined elsewhere and repeated here for the convenience of implementers.

application/ocsp-response
OCSP Response token as specified in [!RFC6090].

application/pkix-cert
A single DER encoded PKIX Certificate as specified in [!RFC5280].

TBS
A Certificate Transparency notary chain as specified in [~RFC6962].

application/pkcs-12
A PKCS#12 encrypted private key.

6. Acknowledgements

Rob Stradling, Robin Alden...

7. Security Considerations

7.1. Denial of Service

7.2. Breach of Trust

7.3. Coercion

8. IANA Considerations

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

9. References

9.1. Normative References

[RFC6960] Santesson, S.,Myers, M.,Ankney, R.,Malpani, A.,Galperin, S.,Adams, C., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, June 2013.

[RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[RFC5280] Cooper, D.,Santesson, S.,Farrell, S.,Boeyen, S.,Housley, R.,Polk, W., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC6090] McGrew, D.,Igoe, K.,Salter, M., "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011.

[I-D.hallambaker-omnibroker] Hallam-Baker, P, "OmniBroker Protocol", Internet-Draft draft-hallambaker-omnibroker-07, 21 January 2014.

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

[RFC2986] Nystrom, M.,Kaliski, B., "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.

[I-D.hallambaker-wsconnect] Hallam-Baker, P, "JSON Service Connect (JCX) Protocol", Internet-Draft draft-hallambaker-wsconnect-05, 21 January 2014.

9.2. Informative References

[RFC6962] Laurie, B.,Langley, A.,Kasper, E., "Certificate Transparency", RFC 6962, June 2013.

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

philliph@comodo.com