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.
# Table of Contents

1. Definitions .................................. 3
   1.1. Requirements Language ................ 3
2. Introduction ................................ 3
   2.1. Traditional Server Configuration Approach .... 3
   2.2. Automating network management. .......... 3
      2.2.1. Cloud Computing Requirements. ....... 4
3. Omnibroker Publication Service ............... 4
   3.1. Service Binding ........................ 5
   3.2. Acquiring Cryptographic Credentials .... 5
      3.2.1. Example: Small Web Site Operator ... 5
      3.2.2. Example: Large Enterprise .......... 10
   3.3. Generating or Obtaining a Public/Private KeyPair. 11
      3.3.1. Example: Internet Coffee Pot ....... 11
   3.4. Request Network Configuration ........... 14
      3.4.1. Example: Coffe Pot Service Registration.. 14
4. OBPPublish ................................ 16
   4.1. OBPPublish Transactions ......... 16
      4.1.1. Advertise ........................ 16
      4.1.2. Credential ........................ 16
      4.1.3. Notify ........................... 16
   4.2. OBPPublish Messages ................ 16
      4.2.1. Message: AdvertiseRequest .......... 16
      4.2.2. Message: AdvertiseResponse ....... 17
      4.2.3. Message: CredentialRequest ....... 17
      4.2.4. Message: CredentialResponse .. 17
      4.2.5. Message: NotifyRequest ........ 18
      4.2.6. Message: NotifyResponse ....... 18
   4.3. OBPPublish Structures ............ 18
      4.3.1. Structure: TaggedBinary ........ 18
5. Transport Bindings and Identifiers ........... 19
   5.1. Content-Type Identifiers ................ 19
6. Acknowledgements .......................... 19
7. Security Considerations .................. 19
   7.1. Denial of Service ................ 19
   7.2. Breach of Trust ................ 19
   7.3. Coercion ................................ 19
8. IANA Considerations ........................ 19
9. References ................................ 20
   9.1. Normative References ................ 20
   9.2. Informative References ............ 20
Author’s Address ............................ 20
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 Omnibroker service locally. The local service would then be configured to forward publication data to any IT facilities 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": "4m7Lzr7g2Fzli1XcVGeDlOw"}

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": "98RV4Se7VQIP3FbqcrLkynUth5u6F48dbCGpzzHpkGfQ",
    "Cryptographic": {
      "Secret": "e6oSWl3dFfnkpYXvSTvYlw",
      "Encryption": "A128CBC",
      "Authentication": "HS256",
      "Ticket": "V8ae-8uQMt_uyKJQb4umJEpys--OXVriEHRoq5sw5uq6u1_4twv8ro7DyD5SuhsPo1bX2c0nD00HS0JpcA1Gs9WjRarqzz0WrD0Inl39d89zbcWoMKYKh1OFFV_LFV8kPPoK8BmaQ0xCo3kBrxg"}
  }
}

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=J5wXEchUbr7k2A0mva0EPn3KGFagzglvH_MX6W1R14;
Id=V8ae-8uQMtt_uyKJQb4umJEpys--OXVriEHRoq5sw5uq6u1_4twv8ro7DyD5SuhsPo1bX2c0nD00HS0JpcA1Gs9WjRarqzz0WrD0Inl39d89zbcWoMKYKh1OFFV_LFV8kPPoK8BmaQ0xCo3kBrxg
Host: localhost:8080
Content-Length: 129
Expect: 100-continue
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

{
   "TicketRequest": {
      "Service": ["omni-publish"],
      "ChallengeResponse": "49GCx5HUvU2SNE6M3GuRcxgFfvZKDLpTfpqXUOAGXVE"
   }
}

{  
   "TicketResponse": {   
      "Status": 200,
      "StatusDescription": "Success",
      "Cryptographic": [{
         "Protocol": "sxs-connect",
         "Secret": "DOdZw6ynGANAjqnR-1gL0A",
         "Encryption": "A128CBC",
         "Authentication": "HS256",
         "Ticket": "Ay9sUcNccO0GH9cY5IrFcrqjFLo6TqTn69u99SUOVpV_xqB05yJ_fLqeI622H-bBu_k1Dhbl1pUGwNAMaGfNWRKgsu97Dc38WBfUyet0M0TwYY"},
      { "Service": ["omni-publish"],
         "Name": "localhost",
         "Port": 8080,
         "Priority": 100,
         "Weight": 100,
         "Transport": "HTTP",
         "Cryptographic": {   
            "Secret": "LfRHVDFWVkVQu81I2wT8w",
            "Encryption": "A128CBC",
            "Authentication": "HS256T128",
            "Ticket": "OufJWKzCHmCLVeSuS4Nh4ozUrfyDa4v8Dd5FIRkIWFQrGnHlG_6ZmoHQQgqIRg7BL7Gm9Jq7LQihkCLhgQjp1LJqDgMxDuDFbH52RD17_g"},
      { "Service": ["omni-publish"],
         "Name": "localhost",
         "Port": 9090,
         "Priority": 100,
         "Weight": 100,
         "Transport": "UDP",
         "Cryptographic": {   
            "Secret": "e10DZJePf2UDWW1hHw2-3A",
      }  
   }  
}  

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-I6gdZPC5ej1;
   Id=OufJWkZCHmCLVeSuS4Nth4ozUrfyDa4v8Dd5FIrkIWFqRgHlNg_62moHqG
   qIRg7BL7Gm9Jq7LQi8kCllhgQjpoLJqgDmDfH5ZRD17_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=ArwY9yiA0aMjxTx4jE_BzNTNJ5z4Nn-16gjdZPC5ejI;
Id=OufJWkZCHmCIVeSu4Nh4ozUrRfyDa4v8Dd5Frkp1WFQrGnHLqG_62m0HQG
qIRg7BL7Gm9Jq7LQihkCLhgsQjp1LJqp0DuDFbH52RD17_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=vVuUnw2Pi0x1HB2OFnOBTDmln9qC1HhEjUkJMhfcfBZ2RCPXc1GzPw8TLm1b8asSGCtD8B681WvoGW0DcEgNMDUc0Uzu-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 suitable 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=tKs0Y7AGoqfFtZDIU395TibIU2E6ciO2beA41nBtgs;
   Id=vVuUnw2Pi0x1HB2OFnOBTDm1n9qc1HhEjUkJMHfCtB2RCPXc1gPw8TLM1b
8asS-GCtdD8H681WvoGW0DcEgNMDUc0UZu-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": 
            eA0Q1vK3hMNUO3d8IXECchavHU1tR501LUnL ccpGL3XRSc1D_3GtNpEwqexojb
By4SywEExzCpGpRzyFiooPBIFk2yrz_ae74YxRyc4OuFryF9CqrshEMli-19Szipl
Lr6_NDwifIMAUH4KZwje6TyVCh0xMHWtYY6T_iitwbhOdxsSxITn4xBEUEZQ3w
mJSy5pRbhuoaOTJ_vexy87eYimn-nKr39w99bVnFwMmPvVmWmMgMF_OxGwpf4Y_2
S4Bko6jphQY6MAC_jjSBxzRKP2FG1JNCE-10I1Oohdtt73z1i1p9nYb0opPL1pmj2
6uXHRY08hy0ewXEdHF_zg"
      },
      "Support": [{
         "ContentType": "application/pkix-cert",
         "Data": "4IvLI64hE-6_3xpMD_GUKyJLjG1q3Hz9i7G8kOJkn6ksjqqWmy_MGLN57dDGnK7D"}}]
null
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 incoming 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: Coffee 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=cWUCjw6DTbyS8o1jkKSfDsJWb_8icI_H7b1Tpb80FJo;
Id=vVuUnw2P10x1HB20FnbOBTDmL9qC1HhEjUkJM/HfCtB2RCPXc1GzPw8Tlmc
8asS-GCtd8H681wvoGW0DcEcNMDUc0UZu-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"}]
    },
    "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=cWUCjw6DTbyS8o1jkKSfDsJWb_8icI_H7b1Tpb80FJo;
Id=vVuUnw2P10x1HB20FnbOBTDmL9qC1HhEjUkJM/HfCtB2RCPXc1GzPw8Tlmc
8asS-GCtd8H681wvoGW0DcEcNMDUc0UZu-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"}]
    },
    "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


9.2. Informative References


Author’s Address

Phillip Hallam-Baker
Comodo Group Inc.

philliph@comodo.com