Outbound Contact Deployment Guide

Pre-Dial Validation

5/9/2025

Contents

Pre-Dial Validation

This topic provides an overview of the pre-dial validation feature that is introduced in Outbound Contact 8.0.001. It describes how to provision and implement pre-dial validation of the dialing records over secure and non-secure HTTP connections.

Pre-Dial Validation Over Non-Secure Connection

Pre-dial validation is an optional first step in the processing of the record. With pre-dial validation in place, OCS connects to the specific Web or Application Server using the HTTP or HTTPS and delivers a specifically formed HTTP `POST` request for each record before dialing this record. The body of the HTTP `POST` request contains properties of the individual record, just as an outbound call contains record-specific attached data. Based on the information received in HTTP `POST` request, Web or Application Server makes a decision whether or not the record should be dialed, and replies to OCS with either an HTTP `200 OK` or a `409 Conflict` response. As a next step in processing, OCS dials the records that were validated successfully (`200 OK` response) and attempts to apply treatments (without dialing) to those records that were validated negatively (`409 Conflict` response). Only those records that have successfully passed the validation are dialed by OCS.

Provisioning

Pre-dial validation is controlled by a number of OCS options, which can be set at various configuration levels. Unless the options are set correctly, pre-dial validation will not take place. By default, OCS behaves consistently with the previous release where pre-dial validation is not possible.

Configure the pre-dial validation feature by using the following options:

- pre-dial-validation
- validation-uri
- validation-timeout-call-result
- http-response-timeout
- http-connection-pool-size

Recommendations for Configuration

The pool size and HTTP request handling timeout (**http-connection-pool-size** and **http-response-timeout** options, respectively) must be set in accordance with the anticipated load on Web or Application Server. OCS will reuse open connections (although request pipelining is not supported by OCS) and will never exceed the connection pool size. After a connection has been opened, OCS will not attempt to close it and will reuse it as long as it's available (no request which has not yet been responded to is currently submitted for this connection) and valid (Web or Application Server has not closed this connection). In case an HTTP request needs to be sent out and no available (spare) connection is found in the pool and no new connection can be opened, OCS will queue the request internally until either the spare connection appears or until the handling timeout for the given request expires. Whenever the request is queued internally, log message 93100 is produced (see also Specific Log Messages for log messages description). When the request completes due to the timeout, log message 93202 is generated (see also Specific Log Messages for log messages description). The presence of 93100 and/or 93202 messages indicates that the setting is too low for the timeout, a slow responsiveness on Web or Application Server side, and/or insufficient connection pool size.

OCS creates a separate connection pool for each `host:port` pair it needs to maintain a connection to. This means that HTTP and HTTPS types of the connections will have separate connection pools, even if the host name is the same for both (this is due to different port numbers, 80 and 443 [defaults]). Short and fully qualified domain names for the host are also qualified as separate hosts by OCS; for example `host1`and `host1.subdomain1.domain1.com` are treated as different hosts by OCS and will therefore be assigned separate connection pools.

Specific Log Messages

OCS logs the following standard messages when processing connections to Web or Application Server:

2102	(data send error)
4500	(connecting)
4501	(server contacted)
4502	(cannot connect)
4503	(connected)
4504	(connection lost)
4541	(message received)

In addition to the standard messages, OCS also logs the result of the pre-dial validation for each dial attempt (all pre-dial validation results messages are of Trace Level):

93200	Pre-dial check completed with positive result for phone <phone number>
93201	Pre-dial check completed with negative result for phone <phone number>
93202	Pre-dial check aborted (timeout elapsed) for phone <phone number>

Whenever the connection pool limit is reached the following Standard Level message is logged:

93100	Maximum connections limit <number of connection> reached for server <Chost:port>

Pre-Dial Validation Protocol Description

This section provides a description of the pre-dial validation HTTP requests and responses.

Requests for Pre-Dial Validation

OCS delivers POST request to Web or Application Server that contains BODY in the application/json format (JavaScript Object Notation). This BODY holds all key-value pairs of the record subject to validation in the same fashion that the outbound call produced by OCS contains key-value pairs in its attached data. Similarly to the outbound call, OCS packs in the JSON BODY some mandatory key-value pairs (those whose keys are prefixed with GSW_) and all user-defined fields configured for delivery using the **send_attribute** field-level option.

Example: Validation Request with BODY

The following example is a request with the BODY (mandatory fields begin with GSW_ and user-defined fields begin with USR_):

```
POST /validation/validate.php HTTP/1.1
Accept: */*
User-Agent: OCS/8.0.001
Host: host1.domain1:80
ETag: 390
Content-type: application/json
Content-length:720

        {
                "GSW_TZ_OFFSET":0,
                "GSW_PHONE":"01282663420",
                "GSW_CALLING_LIST":"PFR_CL_01",
                "GSW_CAMPAIGN_NAME":"Campaign One",
                ...
                "GSW_RECORD_HANDLE":390,
                ...
                "GSW_CALL_ATTEMPT_GUID":"00S0VQKQK0DHT2O518838SDAES000098",
                "USR_FIELD1": "John Doe",
                "USR_FIELD2": "501-12-4312",
                ...
                "USR_FIELDT": "2010-05-06 13:25:10.003",
                "USR_FIELDN": 1970
        }
```

Notice that the ETag header of the request is always present and holds the value of record handle of the record being populated.

Processing the Pre-Dial Validation Request

Web or Application Server needs to make a decision about whether or not the record that has been delivered to it in the POST request by OCS is allowed to be dialed. This decision is made based on the individual record properties passed in the JSON BODY of the POST request, as required by the business logic. Web or Application Server should reply with `200 OK` for a positive validation result and with `409 Conflict` for a negative validation result.

Both positive and negative validation results may contain a BODY that must also be in the application/json format. This BODY can contain mandatory and user-defined fields that are to be updated. For example, a negative validation response might include a call result value that will be assigned to record or a positive validation response might contain the timestamp of the validation attempt.

Positive Response to the Validation Request

The following example is a positive validation response. Notice that the ETag header must be present in the response and must contain the value of the record handle of the record being validated.

```
HTTP/1.1 200 OK
Date: Fri, 30 Apr 2010 19:40:58 GMT
Server: Apache/2.2.14 (Win32) mod_ssl/2.2.14 OpenSSL/0.9.8k PHP/5.2.8
X-Powered-By: PHP/5.2.8
ETag: 390
Content-Length: 112
Content-Type: application/json
        {
                "USR_FIELDT": "2010-05-06 15:00:00.000"
        }
```

As a result of this response received by OCS, the record with a record handle `390` will be dialed. The user-defined field of the record with the **send_attribute** option set to `USR_FIELDT` will be updated in the Calling List table with the new string value `"2010-05-06 15:00:00.000".` Notice that the `BODY` part of the positive response is optional and should be provided only if some fields of the record require an update.

Negative Response to the Validation Request

The following is an example of a negative validation response. Notice that the ETag header must be present in the response and must contain the value of the record handle of the record being validated.

```
HTTP/1.1 409 Conflict
Date: Fri, 30 Apr 2010 19:40:58 GMT
Server: Apache/2.2.14 (Win32) mod_ssl/2.2.14 OpenSSL/0.9.8k PHP/5.2.8
X-Powered-By: PHP/5.2.8
ETag: 390
Content-Length: 62
Content-Type: application/json
        {
                "GSW_CALL_RESULT":53,
                "USR_FIELDT": "2010-05-06 15:00:00.000"
        }
```

As a result of this response received by OCS, the record with a record handle `390` will be marked with call result `53` (Wrong Number) and will not be dialed; the user-defined field of the record with the **send_attribute** option set to `USR_FIELDT` will be updated in the Calling List table with the new string value `"2010-05-06 15:00:00.000".` After assigning call result `53` to the record, OCS attempts to apply the treatment to specified call result (if such treatment is configured). Notice, that the `BODY` part of the response is also optional. If no call result is explicitly specified, call result `52` (Cancel Record) is applied.

Timeout While Processing the Validation Request

It is possible that the Web or Application Server will not be able to handle a validation request during the time period that is specified by using the **http-response-timeout** option. In this situation, the validation request will be aborted, and OCS will treat the timeout situation as a negative validation outcome (no dialing will take place). OCS will apply the call result that is specified in the **validation-timeout-call-result** option (default, 3 [General Error]) and will attempt to apply the treatment to that call result.

Pre-Dial Validation Over Secure Connection

This section describes pre-dial validation over a secure connection, including information about provisioning and secure connection-specific log messages.

Preface

For pre-dial validation, OCS supports communication over HTTPS, or strictly speaking, HTTP over Transport Layer Security (TLS) connection, using a Genesys TLS implementation. For a detailed description of a Genesys TLS implementation, see the *Genesys Security Deployment Guide*.

On Windows platforms, support for SSL/TLS is integrated into the operating system.

On UNIX, Genesys Security Pack must be installed on the OCS host. See the Installing Genesys Security Pack in the *Genesys Security Deployment Guide*.

There are two different ways to configure a secure connection:

- Simple TLS connection when only a server certificate is validated.
- Mutual TLS connection when both server and client certificates are validated.

Provisioning for Simple TLS Connection

1. The protocol (scheme) part of the URI that is specified in the **validation-uri** option must have a value of `https://`. In accordance with HTTPS definitions, OCS will connect to port 443 (instead of 80) if a port number is not explicitly specified in the URI.
2. The user must create a Host configuration object with the same **Name** property as the host name that is specified in the **validation-uri** option.
3. The issuer certificate chain (the root CA and intermediate CA if there was any) that was used to sign the server-side certificate must be accessible for OCS:
    - (OCS on UNIX) The path to the issuer certificate chain file (see Supplying Multiple Certificate PEM Files in the *Genesys Security Deployment Guide* for details) must be specified in the **Trusted CA** field of the Network Security section of a Host configuration object mentioned above. An issuer certificate chain file must be Privacy Enhanced Mail (PEM)-encoded, accessible, and have correct access rights.
    - (OCS on Windows) The issuer certificate chain must be installed into the **Trusted Root Certificate Authorities** directory under the **Local Machine** storage on the OCS host.
4. The **Common Name** property of the server-side certificate must match the corresponding host name.

Provisioning for Mutual TLS Connection

1. The protocol (scheme) part of the URI that is specified in the **validation-uri** option must have a value of `https://`. In accordance with HTTPS definitions, OCS will connect to port 443 (instead of 80) if a port number is not explicitly specified in the URI.

2. The user must create a Host configuration object with the same **Name** property as the host name that is specified in the **validation-uri** option. The new Host object must have the following configuration option set in the Annex section: **security/tls-mutual**=1.
3. The issuer certificate chains (the root CA and intermediate CA if there was any) that were used to sign both OCS instances and the pre-dial validation service's certificates must be accessible for OCS:
   - (OCS on UNIX) The path to the issuer certificate chain file (see Supplying Multiple Certificate PEM Files in the *Genesys Security Deployment Guide* for details) must be specified in the **Trusted CA** field of the Network Security section of a Host configuration object mentioned above. An issuer certificate chain file must be Privacy Enhanced Mail (PEM)-encoded, accessible, and have correct access rights.
   - (OCS on Windows) The issuer certificate chain must be installed into the **Trusted Root Certificate Authorities** directory under the **Local Machine** storage on the OCS host.
   - An issuer certificate chain file must contain issuer certificate chains for both server and client certificates.
4. The **Common Name** property of the server-side certificate must match the corresponding host name.
5. The client certificate with its private key must be accessible for OCS:
   - (OCS on UNIX) The path to the OCS certificate file must be specified in the **Certificate** field. The path to the OCS certificate private key must be specified in the **Certificate Key** field. The path to the pre-dial validation service CA certificate file, if this CA is not public, must be specified in the **Trusted CA** field in the **Network Security** section of the Host configuration object mentioned above.
   - (OCS on Windows) The OCS certificate with its private key must be installed into the Personal directory under the **Local Machine** storage on the OCS host. The **Certificate** field must be populated with the fingerprint of the OCS certificate.

For more information about configuring `Host` and `Application` objects, see Securing Connections Using TLS in the *Genesys Security Deployment Guide*.

Examples of Mutual TLS Connection Configurations

EXAMPLE 1. OCS configured on the Linux host **ocs-host** connects to the pre-dial validation service configured on the **pdvs-host** host and port **8080** through the mutual TLS using the public CAs for signing both certificates.

1. Set the **default/validation-uri**=`https://pdvs-host:8080` configuration option in the OCS application.
2. Create a Host configuration object with the **pdvs-host** name and the **security/tls-mutual**=1 option configured in Annex.
3. Because both OCS and the pre-dial validation service use public CAs, there is nothing to configure in this step.
4. Ensure that the pre-dial validation services certificate is issued for the **pdvs-host** host.
5. Populate the **Certificate** and **Certificate Key** fields in the OCS application with the paths to corresponding PEM files. For example: Certificate=`/genesys/ocs-certs/cert.pem`, Certificate Key=`/genesys/ocs-certs/cert-priv-key.pem`

EXAMPLE 2. OCS configured on the Windows host **ocs-host** connects to the pre-dial validation service configured on the **pdvs-host** host and port **8080** through the mutual TLS using the public CAs for signing both certificates.

1. Set the **default/validation-uri**=`https://pdvs-host:8080` configuration option in the OCS application.
2. Create a Host configuration object with the **pdvs-host** name and the **security/tls-mutual**=1 option configured in Annex.
3. Because both OCS and the pre-dial validation service use public CAs, there is nothing to configure in this step.
4. Ensure that the pre-dial validation service certificate is issued for the **pdvs-host** host.
5. Install the OCS certificate in the Personal directory under the **Local Machine** storage on the **ocs-host** host. Populate the **Certificate** field in the OCS application with the OCS certificate fingerprint—for example, `5f124fcc249d4b17947aa75083fa8606d5c4da60`.

EXAMPLE 3. OCS configured on the Linux host **ocs-host** connects to the pre-dial validation service configured on the **pdvs-host** host and port **8080** through the mutual TLS using custom CAs for signing both certificates.

1. Set the **default/validation-uri**=`https://pdvs-host:8080` configuration option in the OCS application.
2. Create a Host configuration object with the **pdvs-host** name and the **security/tls-mutual**=1 option configured in Annex.
3. Populate the **Trusted CA** field in the OCS application with the path to a PEM file containing the issuer certificate chains for both OCS and the pre-dial validation service certificates—for example, `/genesys/ocs-certs/full-ca-chain.pem` .
4. Ensure that the pre-dial validation service certificate is issued for the **pdvs-host** host.
5. Populate the **Certificate** and **Certificate Key** fields in the OCS application with the paths to corresponding PEM files—for example, Certificate=`/genesys/ocs-certs/cert.pem`, Certificate Key=`/genesys/ocs-certs/cert-priv-key.pem`

EXAMPLE 4. OCS configured on the Windows host **ocs-host** connects to the pre-dial validation service configured on the **pdvs-host** host and port **8080** through the mutual TLS using custom CAs for signing both certificates.

1. Set the **default/validation-uri**=`https://pdvs-host:8080` configuration option in the OCS application.
2. Create a Host configuration object with the **pdvs-host** name and the **security/tls-mutual**=1 option configured in Annex.
3. Install the issuer certificate chains for both OCS and the pre-dial validation service certificates in the **Trusted Root Certificate Authorities** directory under the **Local Machine** storage on the **ocs-host** host.
4. Ensure that the pre-dial validation services certificate is issued for the **pdvs-host** host.
5. Install the OCS certificate in the Personal directory under the **Local Machine** storage on the **ocs-host** host. Populate the **Certificate** field in the OCS application with the OCS certificate fingerprint—for example, `5f124fcc249d4b17947aa75083fa8606d5c4da60` .

Secure Connection-Specific Log Messages

When a secure connection to the Web or Application Server is established, OCS prints a specific trace-level message into the log output. As shown in the following example, this message contains the properties of used certificates:

```
8103 Secure connection is established. type 'client', info '1600-135.225.58.18:443',
issuer 'C=US, S=California, L=Daly City, O=Genesys, OU=Outbound, CN=Outbound
Certificate Authority.
```

If a secure connection cannot be established, the following standard-level messages are logged:

8100	Certificate is expired
8101	Certificate is not valid
8102	Secure connection error

Troubleshooting HTTPS Connectivity

HTTPS connection with server-side certificate verification

OCS successfully opens an HTTPS connection (server-side certificate verification only) if the conditions described below are satisfied.

To confirm the HTTPS connection, in the OCS application, set the **x-conn-debug-all** option in the **log** section to 1 and look for the following string (with the respective host and port specified) in OCS logs:

```
tls: ("host"="<host name>","port"=<port>,"protocol"="tcp","transport"=("tls"="1"))
```

If there is no such string, then some of the conditions are not satisfied.

To troubleshoot the HTTPS connection, ensure the following:

1. The target resource URI must contain the **https://** prefix.
2. There must be a Host object specified in the Configuration Settings that corresponds to the host specified in the target resource URI:
    - The name of the Host object must be equal to the host name of the target resource URI.
    - The Annex tab (Options tab in GAX) of this Host object must contain the **client-auth** option set to `0` in the **security** section.
    - (Linux only) The **Network Security/Trusted CA** configuration field of this Host must contain the path to the CA PEM certificate of the target resource on the OCS host, if the root CA of the target resource is not a public CA (DigiCert, and so on).
    - The account under which OCS is running must have sufficient access rights to access this Host object.

TLS connection to HTTPS resource

OCS successfully opens a TLS connection to an HTTPS resource if the conditions described below are satisfied. To confirm the secure connection, look for the following string in the OCS logs:

```
Secure connection is established
```

If there is no such string, then some of the conditions are not satisfied.

To troubleshoot the secure connection, run the following command on the OCS host and validate the output (specify **-CAfile** if the root CA of the target resource is not a public CA, and use the same value as specified in the **Network Security/Trusted CA** option of the target resource Host object):

```
openssl s_client -connect <host>:<port> [-CAfile <target resource root CA certificate file
path>]
```

Ensure the following:

- The specified host is accessible and the specified port is opened.
- There is a network connection between the OCS host and the target host.
- OCS should be able to validate the entire certificate chain related to the target resource certificate, which is obtained from the target resource during a TLS handshake.
- The certificate chain obtained from the target resource during the TLS handshake must not contain self-signed certificates, including the certificate of the target resource itself.

Note: When reporting an issue related to the OCS HTTPS connectivity to an external HTTPS resource, provide the entire output of the **openssl s_client** command that is run on the OCS host to Genesys Customer Care.

Example of successful "openssl s_client" output

This is an example of the successful **openssl s_client** output for a resource that provides a certificate signed with a public CA:

```
$ openssl s_client -connect predialvalidation.domain.com:443
CONNECTED(00000184)
---
Certificate chain
 0 s:CN = predialvalidation.domain.com
   i:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
 1 s:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
   i:O = Digital Signature Trust Co., CN = DST Root CA X3
---
Server certificate
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
subject=CN = predialvalidation.domain.com
issuer=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 3158 bytes and written 422 bytes
Verification: OK
...
```

Example of unsuccessful "openssl s_client" output

This is an example of the unsuccessful **openssl s_client** output for a resource that provides a certificate signed with a non-public CA while the root CA certificate is not provided in both the server certificate chain and in the **-CAfile openssl** option:

```
$ openssl s_client -connect predialvalidation.domain.com:443
CONNECTED(0000016C)
---
Certificate chain
```

```
 0 s:CN = predialvalidation.domain.com
   i:CN = Example Test Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
subject=CN = predialvalidation.domain.com
issuer=CN = Example Test Root CA
---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 2425 bytes and written 405 bytes
Verification error: unable to verify the first certificate
...
```

TLS SNI Configuration

If a TLS connection is not established and the following pattern is found in the logs, it indicates that the remote server has a certificate with multiple names and a TLS SNI configuration is required from OCS:

```
@15:13:41.5756 error:14094458:SSL routines:ssl3_read_bytes:tlsv1 unrecognized name
@15:13:41.5756 Additional data: SSL alert number 112
```

To configure TLS SNI, add the **security/tls-target-name=hostname** configuration option to the Annex section of the host object related to the target pre-dial validation host, where **hostname** is the name of a specific target host.