CoreLogic

How to use the Participant Reporting API

This document provides the basics of the Participant Reporting API. If you are comfortable with writing web-based API calls and just want the details, please check the documentation site: https://trestle-documentation.corelogic.com/participant-reporting-api.html

1. Get an Authentication Token.

To use the participant reporting API, you must first get an authentication token. You pass your API or RETS credentials to the authentication endpoint and it returns a special token that you use in future requests to indicate that you are authenticated and logged in. Details of the auth endpoint can be found at:
https://trestle-documentation.corelogic.com/webapi.html#authentication

The participant reporting API uses the same authentication endpoint as our WebAPI and RETS (using OAuth2 authentication) services do. If you already know how to use the authentication endpoint, you can skip to the next section.

Getting an authentication token is as easy as making a simple HTTP request to our server and getting the token out the results. There are a couple of things to pay attention to when making this request:

1. The request must be a POST request. The url is:
   https://api-trestle.corelogic.com/trestle/oidc/connect/token
2. The request must include the header:
   `Content-Type: x-www-form-urlencoded`
3. The body of the request must look like:
   `client_id=your_client_id&client_secret=your_client_secret&grant_type = client_credentials&scope=either_api_or_rets`

For example, if you use Trestle for RETS and your client ID is LukeSkywalker and your client secret is DarthVader06191977, then your POST body would be:
`client_id=LukeSkywalker&client_secret=DarthVader06191977&grant_type=client_credentials&scope=rets`

The authentication endpoint will respond with JSON including the access_token to use in your next step. This token has an 8hr expiration period, which means you can copy and paste it into the next requests by hand instead of parsing the response with code.


```
{
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1Ni...3wQJVBA",
    "expires_in": 28800,
    "token_type": "Bearer"
}
```

2. Craft and send your Participant Reporting submission.

The core of the Participant Report submission is a JSON object, the basic structure of which is:

```
{
"ReportDate": "2023-04-02",
  "ReportFrequency": "Monthly",
  "Items": [
    {
      "OriginatingSystemName": "CRMLS",
      "ParticipantID": null,
      "ParticipantFirstName": "Joe",
      "ParticipantLastName": "Smith",
      "ParticipantEmail": "joesmith@mailinator.com",
      "ParticipantOfficeID": null,
      "ParticipantOffice": null,
      "ParticipantType": "Agent",
      "DisplayURLs": [
        "https://wwww.joesmithhomes.com"
      ],
      "ServiceStartDate": "2019-08-22",
      "ServiceTerminationDate": null,
      "Notes": ""
    }
  ]
}
```

Details about the meaning of the fields you're submitting can be found in the docs, at: https://trestle-documentation.corelogic.com/participant-reporting-api.html

If you are reporting on multiple participants, you can choose to send one request with all of the participants in the `Items` array or send one request per participant, whichever is easier for you.

Supply the authentication token from the last step in an HTTP header like:
`Authentication: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1Ni...3wQJVBA`

Unlike in the first step, you do not need to set a Content-Type for this request.

POST this JSON document to the Participant Reporting endpoint at:

https://api-trestle.corelogic.com/trestle/report/TpParticipantReport.

3. Understanding the result

If your query is built correctly and you are properly authenticated, the server will send back a message saying that your report was accepted. Otherwise, it will provide an error message with details about what specifically is invalid.

A successful submission will get a response like this:


```
{
"@odata.context": "https://api-trestle.corelogic.com/trestle/report/$metadata#TpParticipantReport/$entity

"TpParticipantReportID": 435,
"TpID": 196,
"BusinessProductID": 47,
"DataFeedID": 17,
  "ReportDate": "2023-04-02"
"ReportFrequency": "Monthly",
"TpNotes": null,
"Recieved": "2023-04-13T15:53:49.8458511Z",
"ErrorCode": "OK",
"ErrorDescription": null
}
```


You can store the TpParticipantReportID that was just created for future reconciliation, if you want. You can also query your past reports on this endpoint (details on the docs site).

An error in field validation or anything else will get a descriptive message that helps you understand what went wrong:


```
{
  "error": {

    "code": "",
    "message": "Items[0]:http:/bogus.bad is not a well formed URL",
    "details": [
      {
        "code": "",
        "target": "Items[0]",
        "message": "http:/bogus.bad is not a well formed URL"
      }
    ]
  }
}
```


In this case, the API is reporting that the one of the DisplayURLs values in the first item in the Items array (“Items[0]”) doesn’t seem to be a valid URL.

NOTE: Your reported participants will “roll up” on the MLS’s reporting by reported member. This means It’s safe to retry a user multiple times until it works.