Voyant Partner Client Services

The following document details the available exposed Voyant Partner Client Services, along with usage details. All the following services should be accessed using the OAuth authorization code grant flow to retrieve an access token. The Voyant Integration Guide can be referenced for more details on authorization.

The swagger file for the Partner Client Services can be found in the attached zip file called "*PartnerClient Attachments*" under the file name "*partner_client_swagger.json*", and its contents can be pasted inside the swagger editor in order to interact with the details of the Partner Client services in more depth using the Swagger UI.

Overview:

Root Endpoint: */partnerClient*

Rest Operation	**Endpoint**	**Description**
GET	/search/{partnerClientId}	Searches for clients in Voyant that are linked to a particular integration.
GET	/{voyantClientId}	Returns a particular client record.
POST	/{voyantClientId}/reconcileIds	Handles the linking of Voyant Ids to a matching item id from an external source.
POST	/{voyantClientId}	Updates a particular client record.
POST	/externalUpload	Uploads client data from external source into Voyant
POST	/upload	Uploads client data to a temporary cache for the authenticated user.

GET /search/{partnerClientId}

Description: Allows a partner to search for clients in the Voyant system that are linked to a partner integration by specifying the partnerClientId.

Usage: Can be used in order to retrieve the Voyant Client Id which is needed for many of the subsequent services listed below, including retrieving a client record, which requires the client's Voyant ID. For a client record to be returned in this search, the adviser either needs to successfully use the external upload endpoint below (/externalUpload), fully finish the cached upload process (/upload), or finish one of the two Voyant UI Internal integration processes (Import or Launch in Context). If the chosen process isn't finished, the client search will not return the desired client reference object.

Sample Request:

https://planwithvoyant.com/advisergo/services/rest/partnerClient/search/29574024947204

Sample Response:


```
[
  {
    "id": "string",
    "owner": "string",
    "primaryFirstName": "string",
    "primaryLastName": "string",
    "spouseFirstName": "string",
    "spouseLastName": "string",
    "partnerRefId": "string",
    "spousePartnerRefId": "string",
    "lastModified": "2022-03-10T15:51:58.585Z",
    "dateCreated": "2022-03-10T15:51:58.585Z",
    "passwordProtected": true,
    "externalClientDataStatus": "NEW",
    "externalClientDataType": "NONE",
    "access": "HOME_HIDDEN",
    "advisorUserInfo": [
      {
        "firstName": "string",
        "lastName": "string",
        "username": "string",
        "email": "string"
      }
    ],
    "homeUserInfo": [
      {
        "firstName": "string",
        "lastName": "string",
        "username": "string",
        "email": "string"
      }
    ]
  }
]
```

GET /{voyantClientId}

Description: Fetch an existing client record in Voyant by the voyantClientId that can be obtained by using the above GET /search/{partnerClientId} request.

Usage: This endpoint can be used to fetch a Voyant client record for external usage in a partner's system, or to just check the current status of a client's data. This endpoint will often be preceded by hitting the /search POST endpoint in order to obtain the client with a voyantClientId:

1. Call GET /search/{partnerRefId} to find the voyantClientId
2. **Fetch the client using GET /{voyantClientId}**

Sample Request:

https://planwithvoyant.com/voyant/services/rest/partnerClient/7028?planId=123&password=789

Query Params:

- ***planId***: *The specific plan id for the client, the default is base or active plan*
  - String
  - Optional

- ***password***: *Password value for password protected clients*
  - String
  - Optional

Sample Response:

A sample client response can be found in the attached zip file under the file name "*client.json*".

POST /{voyantClientId}/reconcileIds

Description: Handles the resolution of ids for duplicate items from different sources. Allows the adviser to reconcile what partnerRefId should be saved for a specific Voyant model to ensure any future update of a client's data from an integration is pointing to the correct source.

Usage: This service should be used to match up items in Voyant that didn't come from any integration source to items that are being passed into Voyant by the POST /{voyantClientId}. This endpoint should be used to prevent duplicate items in Voyant (i.e. an item created in Voyant internally and the same item in the source system, with a different id reference, coming from an update POST /{voyantClientId} body). This endpoint will often be used in the following sequence:

1. Call GET /search/{partnerRefId} to find the voyantClientId
2. Fetch the client using GET /{voyantClientId}
3. Within the calling / source application, process the plan retrieved in Step 2 to determine if any source items need to be mapped to new items in Voyant.
4. **Map said items from Step 3 using the POST /{voyantClientId}/reconcileIds**

Sample Request:

https://planwithvoyant.com/voyant/services/rest/partnerClient/7028/reconcileIds

Body:

- List of Mapped Voyant Model Ids to Partner Reference Ids
  - Required
  - Sample:

```
[
    {
        "id": "7055d62bc0a80125682a580a729c2dd0",
        "partnerRefId": "0F785C8B-1C1A-4E41-9148-CFF225BE0A84"
    },
    {
        "id": "8492e05jo0e39205820q098k238d3go0",
        "partnerRefId": "AC12438D-4E15-4147-943C-02163F5DD451"
    },
    {
        "id": "ef799d4068bc436ca72b01900939ac9f",
        "partnerRefId": "5C539825-19A8-4CB0-A302-C451F9BF0986"
    },
    {
        "id": "08c85278f02b4321853f9b7f9e7d3ca4",
        "partnerRefId": "1CD01328-DD90-4A63-93BE-0191D3C837A5"
    },
    {
        "id": "6c61705ac5e14fbe987ca7e02df089e5",
        "partnerRefId": "2387CB47-54C2-41B4-9B67-9F38F74C3878"
    },
]
```


In this sample request body, the “id” value represents a Voyant Model Id in the Voyant system, and the mapped “partnerRefId” value is the id for the same model, but in a different system.

Sample Response:

Code	*Description*
200	Successful reconciliation of ids.
500	Internal Server Error, check mapping ids.

POST /{voyantClientId}

Description: Update an existing client record in Voyant using the voyantClientId that can be obtained from the /search/{partnerClientId} endpoint.

Usage: This post can be used to make updates to a specific client outside of AdviserGo, while allowing you to see the posted changes the next time an adviser logs back into AdviserGo. Allows for the bypassing of the AdviserGo UI update flow and can be used to update a client's data in a partner's system. The steps below illustrate the sequence of Partner Client Services that will often be used in order to complete an update to this POST /{voyantClientId} endpoint.

1. Call GET /search/{partnerRefId} to find the voyantClientId
2. Fetch the client using GET /{voyantClientId}
3. Within the calling / source application, process the plan retrieved in Step 2 to determine if any source items need to be mapped to new items in Voyant.
4. Map said items from Step 3 using the POST /{voyantClientId}/reconcileIds
5. **Upload new JSON by using this endpoint, POST /{voyantClientId}**

Sample Request:

https://planwithvoyant.com/voyant/services/rest/partnerClient/7028

Body:

- *Client Object*
  - Required
  - Supported Data Formats: json, xml
  - A sample request body for this endpoint can be found in the attached zip file under the file name "*client.json*".

Sample Response:

Code	*Description*
200	Successful update of client details
500	Internal Server Error

POST /externalUpload

Description: Uploads client data into Voyant system directly without using the AdviserGo UI for importing a client.

Usage: This service allows for uploading clients quickly if the adviser has access to the already formatted Voyant client json or xml payloads. Client should not already exist when using this endpoint, if an update on a client's data is desired, the POST /{voyantClientId} endpoint should be used.

Sample Request:

https://planwithvoyant.com/voyant/services/rest/partnerClient/externalUpload

Body:

- Client Object
  - Required
  - Supported Data Formats: json, xml
  - A sample request body for this endpoint can be found in the attached zip file under the file name "*client.json*".

Sample Response:

Code	*Description*
200	Successful upload of client data.
500	Internal Server Error, check POST data format.

POST /upload

Description: This POST /upload endpoint is the first step prior to launching in context if you're looking to create or update a client record and use the Voyant built integration UI flows. Following a call to this endpoint, the calling application should direct a user via the browser to the Voyant launch in context URL with the same specified partnerClientId.

Usage: This usage differs from the above /externalUpload endpoint in that this upload requires the finishing of the launch in context flow through the Voyant UI for the client to be saved in the Voyant system. While this endpoint requires an extra step to complete the saving of a client, it allows the adviser to not include certain client detail items before the client gets saved by using the launch in context flow within the AdviserGo UI. This would be useful if the adviser wants more control over the client info that gets saved in Voyant. The following sequence of steps illustrates in what context this POST /upload service should be used.

1. **Call POST /upload with Client JSON or XML included in the Body of the request**
2. Navigate User via the browser to the Voyant Launch in Context URL with the same partnerRefId used in the Client body.
    a. Launch In Context URL:
    https://planwithvoyant.com/advisergo/launch/family/{partnerRefId}
3. Complete the Launch in Context flow within the AdviserGo UI

Sample Request:

https://planwithvoyant.com/voyant/services/rest/partnerClient/upload

Body:

- Client Object
    - Required
    - Supported Data Formats: json, xml
    - A sample request body for this endpoint can be found in the attached zip file under the file name "*client.json*".

Sample Response:

Code	*Description*
200	Successful upload of client data.
400	Request Body was not complete, check client data format.
500	Internal Server Error, check POST data format.