Contents

1. Introduction

For Business Intelligence (BI) dashboards, a technical design is in the works. However, a few customers are more likely to want to get started with Power BI and develop dashboards on their own.

Within the technical design, a data model was created for a database containing copy data from Snagstream. This data model is designed to:

- The naming of master data in Snagstream;
  The naming of the entities in the API does not align with the naming of them in the Snagstream front-end and the Snagstream App. In the data model, this is the case.

- Requesting data on answered questions and posted snags.
  The Snagstream database itself is focused on flexibility in terms of provisioning and working the data of a snag or a form within a project.
  The BI data model is aimed at requesting data about answered questions, whether or not within a certain project.

In this document, API calls (end-points) that support the current version of Snagstream are related to the data model. The aim of this document is to support customers so that they can independently set up a BI database with Api calls and, for example, develop dashboards with Power BI.

2. Api requests

The Snagstream API is secured with Bearer Authentication. To access your environment, first request a token from Pro4all. This token contains a user account. It is recommended to create a new user with a name that indicates that it is the API user (for example, api@snagstream.nl).

After you receive a token, you can use the API. In all requests, the token must be sent in an http authorization header:

Authorization: Bearer <your token>

The API works on the basis of Json data. Therefore, send the following http accept header with a Get command (this is optional):

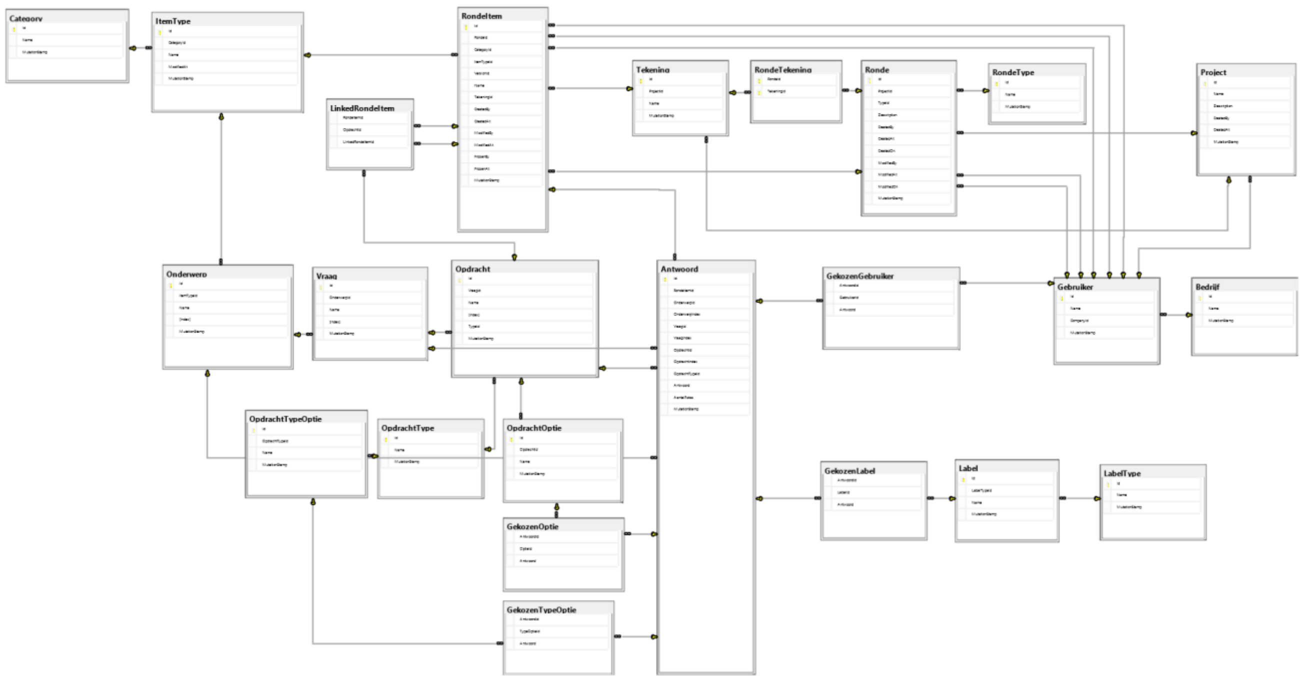
Accept: application/json

and for a Post or Put command, the following content-type header:

Content-Type: application/json

3. The data model

The following figure shows the complete data model. This model is then zoomed in in separate paragraphs.



3.1 Data model relationships with the Snagstream front-end

The following table shows per table from the data model where it comes back in Snagstream web.

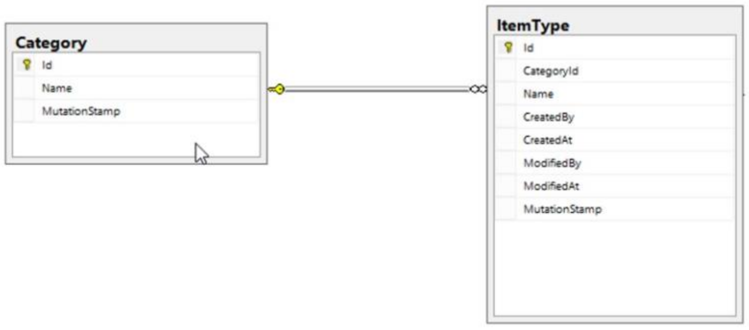
Table	Front-end
Category	This table contains "fixed" data. Categories cannot be mutated from the front-end. There are 2 categories: • Snag • Form
Item Type	This is a Snag or a Form. A reference to the Category table determines whether the ItemType is a Snag or a form. ItemType are created by creating a new snag or form via the following front-end route: • Management | Templates | Snags • Management | Templates | Forms
Subject	A Snag or Form is divided into Topics. Topics are created from the Snag or Form screen.
Question	A topic within a Snag or form is divided into questions. Questions are created from the Snag or Form screen.
Commission	A question within a Snag or form is divided into assignments. Commands are created from the Snag or Form screen.
Command Option	A list of options can be linked to commands of the following type: • Selection • Buttons • Action • Multiple choice • Priority Command options are created from the Snag or Form screen.
Command Type	The Assignment type determines the "type of answer" (employee choice, Status choice, etc.). Command types cannot be mutated from the front-end. There is a fixed list of Command types.

Command Type Option	The following command types have a fixed list of options associated with them • Agreement • Snag priority (priority default) • Status These options cannot be mutated from the front-end. There is a fixed list of options per type. The following CommandType can be done from Administration | Type/Labels a list of Types and related Labels are entered. • Type/Tag To do DETERMINE WHERE THESE ARE IN THE MODEL
	Types and Tags are written in the ItemModuleTypeOption table. Types do not have parentId and Tags/Labels do, and it refers to the Type.
LabelType	Label types are created and mutated from the front-end path: Management | Type / Label
Label	Label types are created and mutated from the label type screen A LabelType and within it a Label can be chosen from the command of the CommandType Type/Label.
User	Users are created and mutated from the front-end path: Management | Users
Company	Companies are created and mutated from the User screen.
Round Type	Round types are created and mutated from the front-end path: Management | Round types
Project	Projects are created and mutated from the front-end path: Management | Projects
Round	Rounds are created from and mutated from the front-end path: • Project | Rounds
Drawing	Drawings are created and mutated from the front-end path: • Project | Drawings

Round Drawing	A drawing can be linked to a round one from the Round screen
Round Item	A round item is a snag or a form linked to a round.
Answer	A Round item consists of a list of questions. While answering these questions, the answers are stored. An answer can be a free text, a date, one or more chosen employees, a choice from a list of possible answers or a photo. Questions are answered from a Snag or Form linked to a project.
LinkedRoundItem	A Snag can be linked to an answer from the form screen. This results in a LinkedRoundItem.

3.2 Item Types and Categories

Within Snagstream you can work with Snags and Forms. In terms of data model, both are the same, only a snag refers to a "snag" Category and a form to a "survey" Category. The following figure shows the data model for Category and ItemType:



3.2.1 Category

Categories can be queried with The following end-point:

GET /api/item/categories/all

This call provides a list of categories with the following properties per Category (not all properties are displayed):

```
{
            "id": "458ac9fe-f77c-4cc1-b47f-d5f857413ac1",
             "Name": "snagCategory"
}
```

3.2.2 ItemType

A list of all ItemTypes can be requested with The following end-point:

POST api/item/categoriesandtypes/all

This call returns a list containing the following information per category:

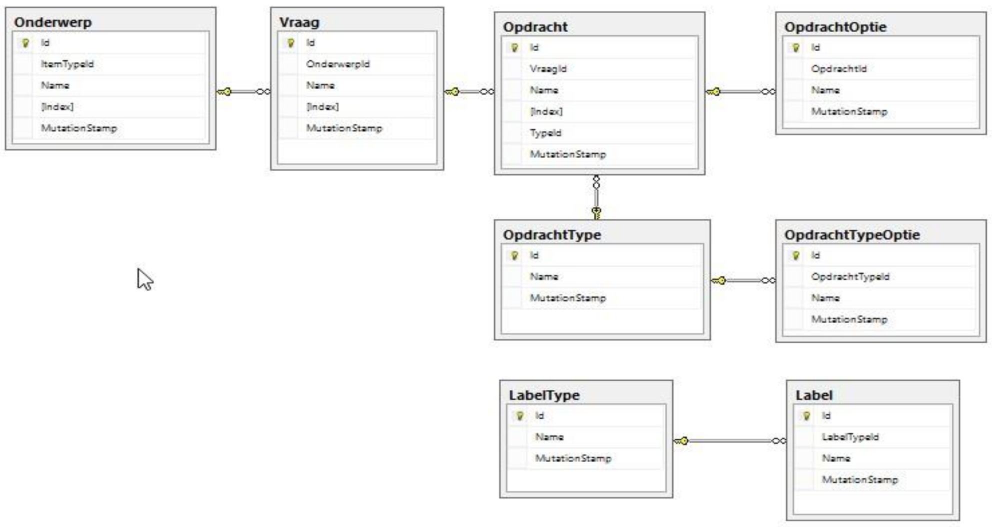
```
{
        "Category": { see the api/item/categories/all call },
          "Types": [ { type } ]
}
```

The types associated with a category are returned as an array of types. The data of a type looks like this (not all properties are displayed):

```
{
        "Type": {
                "Id": "b02fde01-d9e2-46c3-a08f-c031b3843d0f",
                "ItemCategoryId": "458ac9fe-f77c-4cc1-b47f-d5f857413ac1",
                "Name": "Inspectionnag",
                 "CreatedAt": "2018-10-17T11:15:42.461649",
                "CreatedBy": "5ce18cc0-d26c-470d-b489-a9ea70df5529",
                 "ModifiedAt": "2020-04-22T09:06:52.281987",
                "ModifiedBy": "5ce18cc0-d26c-470d-b489-a9ea70df5529",
                  "Versions": [{
                          "id": "1019a8ea-d459-443e-9441-67f52cc45a0a",
                          "CreatedAt": "2020-04-22T09:06:52.382271",
                          "CreatedBy": "5ce18cc0-d26c-470d-b489-a9ea70df5529"
                  }]
        },
          "Versions": [{ same list as above } ]
}
```

3.3 Topics, questions and assignments

The following figure shows the Topics, Questions, and Assignments tables with related tables:



An ItemType (Snag or Form) is divided into Topics, Questions, and Assignments. The following call provides the details of an ItemType:

POST api/items/itemDictionary/latest/{itemTypeId}

A filter is set in the body of this call:

```
{
        "itemVersionIds": [null]
}
```

By providing versionId's you can get the details of a specific version of the ItemType. If you do not filter by version IDs, you will receive the latest version of the ItemType.

Note: Although the name suggests otherwise, the response contains information of all versions of the item if no ItemVersionIds are sent in the body.

This call returns a fairly complex structure containing a lot of data duplicately. For the data model, only the list of Versions is important. The complete structure looks like this:

```
{
        "Conditions": [],
        "FieldConditions": [],
        "Fields": [],
        "ModuleOptions": [],
        "Modules": [],
        "ModuleTypes": []
        "Sections": [],
        "Types": [],
        "Versions": []
}
```

The "Versions" property is an array/list "Version" information with the following data per Version:

```
{
         "Id": "c95be59a-b284-44e3-a4be-4d34242726ca",
        "Sections": []
}
```

The data of a Version does not appear in the data model. The list of Sections contains the following information per Section:

```
{
         "Id": "150f7107-7261-4042-a816-853aa9b9b1ac",
        "Index": 1,
         "Name": "General",
        "Fields": []
}
```

A Section is a Topic in the BI data model. The list of Fields contains the following information per Field:

```
{
       "id": "8f904646-801b-481a-a70c-5e4c5bcf2aba",
                "Index": 1,
                "Name": "Description",
                "Modules": []
        }
```

A Field is a Question in the BI data model. The list of Modules contains the following information per Module:

```
        {
                "id": "dc379f47-9cee-4e33-b64d-58e2c04cf6cd",
                "ItemModuleTypeId": "be62d7d5-d5ec-41e0-a57a-747a2b63f4d7",
                "Name": "Description",
                "Index": 1,
                "ModuleOptions": []
        }
```

A Module is an Assignment in the BI data model. The ItemModuleTypeId property refers to an entry in the CommandType table.

The ModuleOptions property can contain an array of ModuleOptions but is empty when a specific version of an ItemType is requested. If the array is filled, it contains the following data per entry:

```
        {
                "Id": "21b1f51d-5623-419a-90d6-b1fcf78a24ab",
                "ParentId": null,
                "Name": "Rounded",
                "ItemModuleTypeId": "7bf16585-70c0-449c-8e28-b6cbef01fb85",
        }
```

A ModuleOption is a Command Option in the BI data model.

The parentId of an entry is populated only if the option is a Type/Label option. The parentId then refers to a LabelType and the Id in the entry itself is the Id of the Label within the LabelType. An example of this is:

```
        {
                "Id": "c2d2658d-8b6c-49fe-bcc1-6de1e52b423b",
                "ParentId": "3bc4a930-1795-4ea6-986b-341c28a7ddd7",
                "Name": "tester de test",
                "ItemModuleTypeId": "0a8d3464-c5dd-4d3a-a1d3-a538225edfd5",
        }
```

The data from the ModleOptions array is inherited into the Command Option table of the data model.

3.3.1.1 Command Types

The following table lists the Command Type IDs that can be used within Snagstream.

CommandTypeId	**Command type**
AB095E22-F97C-4C88-BB4F-CF98675065ED	Multiple choice
91c15760-001f-47f1-a75f-3e02ac003a6d	Selection

BE62D7D5-D5EC-41E0-A57A-747A2B63F4D7	Free field
4F225325-1E2D-4662-8E6D-FA073BAD94A2	Agreement
837a73eb-f637-4b14-b2eb-ca082a06168f	Action
c4df9b2c-d6bb-42fc-a1a8-9caf4ea56556	Users
bfd945df-dab8-4dd0-ae60-6344304a1666	Priority standard
EE2BA447-96B4-44CE-AAE4-605083A525CF	Priority unique
EE5C7106-A6F7-46C0-8834-8F9F1F8A8536	Date
E8B26278-642E-4F4F-AFF9-B97B1CDC2FCF	Number
2481d661-7027-4e0e-a106-98cb2de75d58	Buttons
2132e34a-cbae-4a79-88b4-aa125a0fee98	Photograph
84e0d26c-3808-488e-aab2-72651f1d2bcf	Signature
0a8d3464-c5dd-4d3a-a1d3-a538225edfd5	Type / Tag
7BF16585-70C0-449C-8E28-B6CBEF01FB85	Status
C5D6CC34-4926-462C-9FC6-54BBF0ABCEF8	CC
60619CA6-054D-4E0F-A2F2-E3B78F7A1E89	Assigned to
42efbebf-0397-4652-90c8-4b4de6647113	Description

There is no end-point to retrieve the data from this table.

Depending on the Command Type, there may be a list of options associated with a Command.

3.3.1.2 Default Command Types with Options

For the standard Snagstream Command Types, there are specific end-points to get the options:

- Agreement
  GET api/items/itemModuleTypeOptions/agree
- Priority
  GET api/items/itemModuleTypeOptions/priority
- Status
  GET api/items/itemModuleTypeOptions/status

For the standard Snagstream Command Types, the options can also be retrieved with this endpoint

GET api/items/itemModuleTypeOptions/{itemModuleTypeId}

The response of these end-points is an array of options with the following data per option:

```
{
  "Id": "8eeef73e-f3e3-4ccc-a79c-e41a526cbaa4",
   "ItemModuleTypeId": "4f225325-1e2d-4662-8e6d-fa073bad94a2",
  "Name": "Yes",
        }
```

This data can be processed in the CommandTypeOption table of the data model.

3.3.1.3 Type/Labels Command Type

All Types and labels can be retrieved at the following end-point

GET api/tag/all

The response is an array of LabelTypes with the following data per entry:

```
{
        "TagId": "a86d7218-0cf0-4bb8-be8a-40650a2e5fae",
        "Name": "Finishing carpentry",
        "Children": [ ]
}
```

This data can be processed in the LabelType table of the data model. The Children array contains the following data per entry:

```
{
        "TagId": "b20ddca5-f8b7-4b3d-bff6-27953ae8a088",
        "ParentTagId": "a86d7218-0cf0-4bb8-be8a-40650a2e5fae",
        "Name": "Paneling is missing"
}
```

This data can be processed in the Label table of the data model.

3.3.1.4 Assignment types where a list of options can be entered

In the Commands linked to one of the following Command Types, a list of options can be defined:

- Selection
- Buttons
- Action
- Multiple choice
- Priority

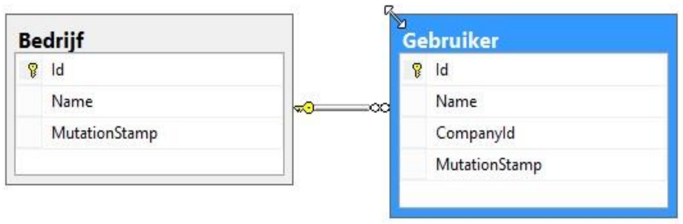
The options associated with a Command options can be accessed from the response to a

POST api/items/itemDictionary/latest/{itemTypeId}

call to be made. See the main paragraph of this sub-paragraph.

3.4 Users

The following figure shows the user table and the directly related tables:



The following end-point lists users:

POST /api/users/allwithdeleted

With in the body:

```
{
  "staticCriteria":[],"
   specificStaticCriteria":[]
}
```

The result is an array of users with the following data per user

```
{
            "UserId": "",
            "CompanyId": "",
            "FirstName":"",
            "SurName","",
            "FullName", ""
}
```

3.5 Do

The following end-point lists companies:

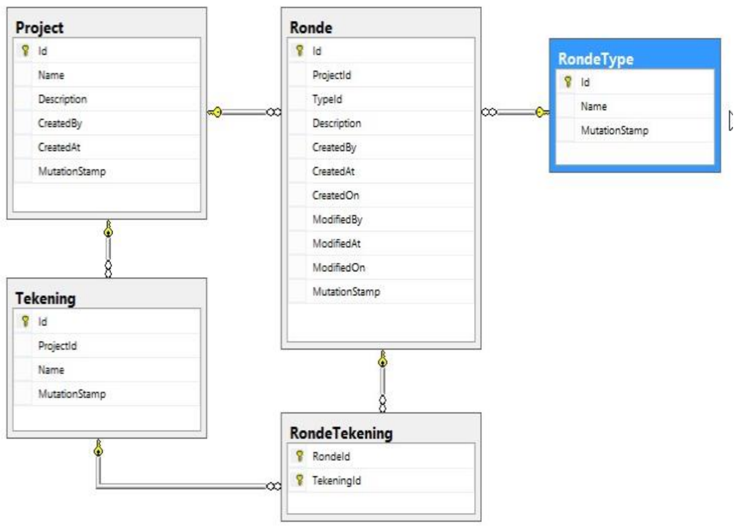
GET /api/company/all

The result is an array of companies with the following data per company

```
{
            "CompanyId": "",
            "Name":""
}
```

3.6 Projects

The following figure shows the projects table and the direct relationships with this table:



The following end-point lists projects:

GET api/projects/all

The result is an array of projects with the following data per project:

```
{
        "ProjectId": "",
        "Name": ""
        "Description": ""
        "CreatedAt": " date time"
        "CreatedBy": " id of a user"
}
```

3.6.1 Project drawings

The following end-point lists drawings associated with a project:

POST api/projects/{projectId}/drawings/all

The result is a list of drawings with per drawing:

```
{
        "DrawingId": "",
        "ProjectId": "reference to the project table"
        "Name": ""
}
```

3.6.2 Project rounds

The following end-point lists rounds associated with a project:
api/visit/{projectId}/all

Lists the rounds of the project with per round:

```
{
        "VisiId": "",
        "ProjectId": " refers to the Project table",
        "Description": "",
        "VisitTypeId": " refers to the RoundType table"
        "CreatedAt": "date time",
        "CreatedBy": "refers to the user table",
        "CreatedOn": "Refers to ?? To do ",
        "ModifiedAt": "date time",
        "ModifiedBy": "refers to the user table",
        "ModifiedOn": "Refers to ?? To do ",
        "Drawings": [ list of drawings ],
}
```

The list of drawings contains the drawings linked to the round. Of the detailed data per drawing from the list, only the following property is important for the data model:

```
{
        "DrawingId": "refers to the drawings table"
}
```

3.6.3 RoundTypes

The following end-point lists RondeTypes:

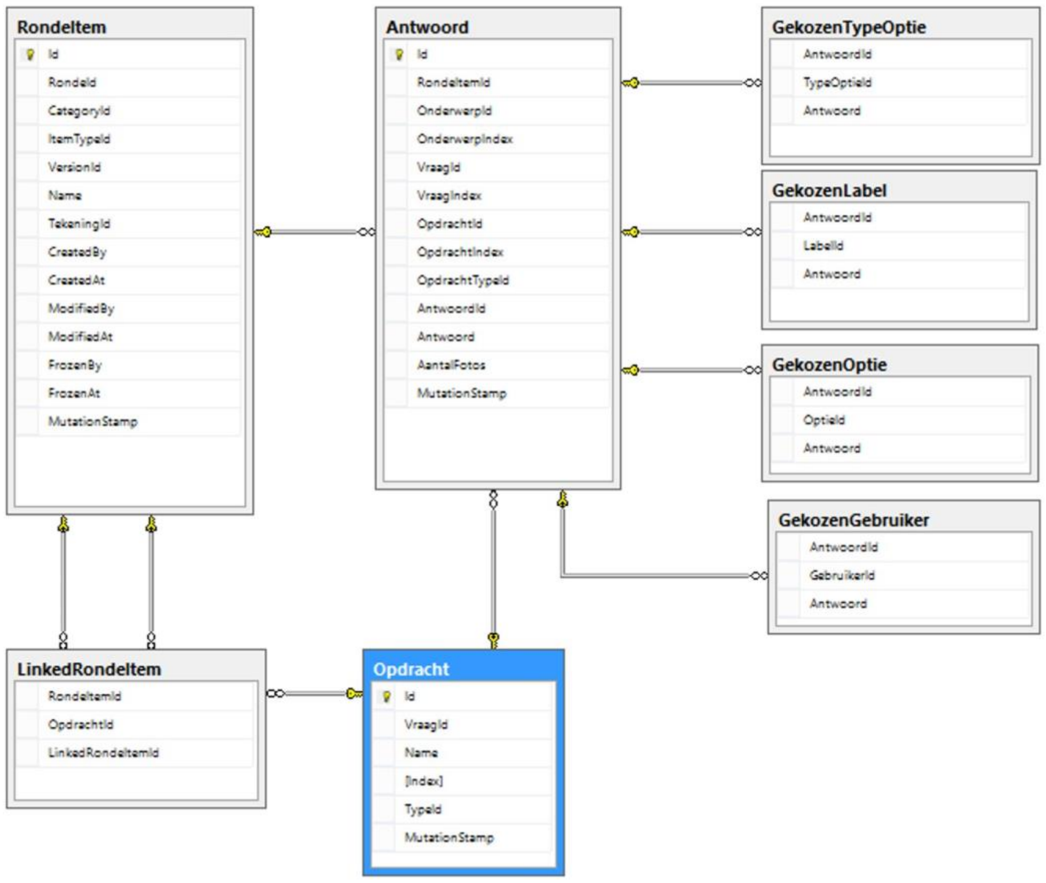
GET api/visit/types

The result is an array of visit types by type containing the following data:

```
{
        "Id": "",
        "Name":""
}
```

3.6.4 Project answers

The following figure shows the tables involved in the response data:



Retrieving answers must be done in 2 steps:

1. Get the IDs from the snags or forms associated with the project
2. Get the answers from the snags or forms.

3.6.4.1 Project snags or form requests

The next end-point

POST api/item/{projectId}/data/lazy/{categoryId}/{itemTypeId}/{itemVersionId}

Displays the snags or forms of a project.

The body of the call specifies the page and the number of lines per page for the response:

```
{
        "offset":0,
        "increment":100
}
```

In the example above, the first page containing 100 lines/items is requested.

The parameters in the path of the call determine what you request:

categoryId	**Kind**
458AC9FE-F77C-4CC1-B47F-D5F857413AC1	Snags
CCB72A59-032D-4867-A2EA-D8AB830B54A1	Forms

itemTypeId	**Kind**
00000000-0000-0000-0000-000000000000	Everything snags or forms linked to the project. Does not contain response data
Specific guid	A specific ItemType (snag or form). Does include response data

itemVersionId	**Kind**
00000000-0000-0000-0000-000000000000	All versions of the ItemType
Specific guid	A specific version of the ItemType

The snags linked to a project are requested with the following call:

POST api/item/{projectId}/data/lazy/458ac9fe-f77c-4cc1-b47f-d5f857413ac1/00000000-00000000-0000-000000000000/0000000-0000-0000-0000-0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

with the pagination parameters in the body.

This gives a list of linked snags. This list is preceded by pagination and filtering information. The overall response is as follows

```
{
        "Offset": 0,
        "Increment": 100,
    "SortOnColumn": "00000000-0000-0000-0000-000000000000000000",
        ....
        "Items": []
}
```

The Items array contains the list of associated Snag/Forms (depending on the categoryId). This list contains per entry in addition to all kinds of detailed information: the Id of the snag/ the form within the project and the ItemTypeId of the item:

```
{
```

```
            ...
            "Id": "03167957-8263-40d8-b2fb-8d14bac75db8",
            "ItemTypeId": "c8f28c00-2c73-4502-925f-
            fde097d85eee", ...
    }
```

The data per Item does not contain response information. This information can be retrieved in two ways.

3.6.4.2 Snags or forms answers by snag or form

This end-point can be called per Item entry from the result of the call from the previous paragraph

POST api/item/{projectId}/data/lazy/458ac9fe-f77c-4cc1-b47fd5f857413ac1/{itemTypeId}/00000000-0000-0000-0000-000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

with the pagination parameters in the body.

By entering an ItemTypeId in the above end-point (c8f28c00-2c73-4502-925f-fde097d85eee in the previous example)) the response data of the relevant snag or form is retrieved.

The result of this call contains the following information:

```
    {
            "Offset": 0,
            "Increment": 100,
        "SortOnColumn": "00000000-0000-0000-0000-000000000000000000",
            ....
            "Items": []
    }
```

with per Item:

```
    {
            "Id": "03167957-8263-40d8-b2fb-8d14bac75db8",
             "VisitId": "150962d0-e6b5-4841-b9fd-0cff6e52ae51",
            "ItemCategoryId": "458ac9fe-f77c-4cc1-b47f-d5f857413ac1",
            "ItemTypeId": "c8f28c00-2c73-4502-925f-fde097d85eee",
           "ItemTypeName": "Snaggy",
          "ItemVersionId": "d3a389fc-29ef-4478-8b3a-2a19632d8260",
             "DrawingId": null,
             "CreatedAt": "2020-04-30T13:36:56.086857",
             "CreatedBy": "5ce18cc0-d26c-470d-b489-a9ea70df5529",
             "CreatedOn": "00000000-0000-0000-0000-
            0000000000000000000000000000000000000000000000000000000000000000000000000000
            00000000000000000000
             "ModifiedAt": "2020-04-30T13:37:16.117427",
             "ModifiedBy": "5ce18cc0-d26c-470d-b489-a9ea70df5529",
             "FrozenAt": null,
             "FrozenBy": null,
             "ModuleInstanceDictionary": { }
     }
```

This data is processed in the RondeItem table:

Repsonian property	**RoundItem column**
Item.Id	Id
Item.VisitId	RondeId
Item. ItemCategoryId	CategoryId
Item. ItemTypeId	ItemTypeId
Item. ItemTypeName	Name
Item. ItemVersionId	VersionId
Item. DrawingId	DrawingId
Item.CreatedAt	CreatedBy
Item.CreatedBy	CreatedAt
Item.ModifiedBy	ModifiedBy
Item.ModifiedAt	ModifiedAt
Item.FrozenAt	FrozenBy
Item.FrozenBy	FrownAt

The ModuleInstanceDictionary property of an item contains a list of ModuleInstance objects. These objects contain the answers. Each object therefore has a line from the response table. This only applies to objects whose Value property is not empty. Only then is an answer entered with the command in Snagstream.

Each object contains the following information:

```
        "ItemModuleViId" : {
                "Id": "e0c33a13-feac-4a1e-9337-1a437764733f",
                "ItemModuleTypeId": "7bf16585-70c0-449c-8e28-b6cbef01fb85",
                "ItemModuleId": "51545010-b870-46eb-826a-cf60a8a83ec3",
                "Value": "aabc7c19-366d-4944-b66e-20ac746fd33c",
                "StringValue": "Feedback desired",
                "Media": []
        }
```

The response data is processed in the Response table and depending on the ItemModuleTypeId in one of "Chosen" tables.

The following table shows the relationship between the columns from the Response table and the response data:

Repsonian property	**Reply column**
Item.ModuleInstance.Id	Id
	SubjectId

	SubjectIndex
	QuestionId
	Demand Index
Item. ModuleInstance. ItemModuleId	CommandId
	CommandIndex
ModuleInstance.ItemModuleTypeId	CommandTypeId
Item.ModuleInstance.StringValue	Answer
Number of entries in the Item.ModuleInstance.Media array	QuantityPhotos

As you may have noticed, the SubjectId, SubjectIndex, QuestionId, QuestionIndex, and CommandIndex columns are empty in the table above. These can be filled in by requesting the assignment and the Subject linked to the database. This will then filter by the CommandId.

Depending on the ItemModuleTypeId, the response property "Value" is processed in one of the "Chosen" tables. Dot is only done for the types to which a list of options can be linked. See the relevant paragraph.

Assignment Type	**Selected table**
Selection	ChosenTypeOption
Multiple choice	ChosenTypeOption
Agreement	ChosenOption
Action	ChosenTypeOption
User list	ChosenUser
Priority	ChosenOption
Picker	ChosenTypeOption
Switch	ChosenTypeOption
Type / Tag	ChosenLabel
Snag priority	ChosenTypeOption
Status	ChosenOption
CC	ChosenUser
Assigned to	ChosenUser

The Value property can contain a list of IDs. For each of these IDs, an entry is then created in the respective Chosen Table. The response column of this table can then be determined by reading the "Name" property from the Option table to which the Chosen table is linked. An alternative is to read the answer from the ValueString, but I don't know if the contents of the Value property and the ValueString property are in the same order.

3.6.4.3 Snags or forms answers for a range of items

The end-point response from the preceding paragraph does not contain any information about the structure of the snag or the form. The end-point in this section does contain this data.

With the following end-point, the response data of a series of snags or forms can be requested:

POST /api/items/{projectId}/full

With in the body

```
{
        "itemIds": [ ],
        "loadDefinition": true
}
```

An array of Item Ids can be sent in the body. Byprocessing the Item Ids from the repsonse described in the section "Project snags or forms", all answers from snags or forms associated with a project are requested.

LoadDefinition determines whether the end-point response contains information about the structure of the snags or forms behind the questions. With "loadDefinition": true, the response contains this structure. The response looks like this:

```
[ { item details }]
```

With per item:

```
{
        "Id": "03167957-8263-40d8-b2fb-8d14bac75db8",
        "ItemTypeId": "c8f28c00-2c73-4502-925f-fde097d85eee",
        "ItemTypeName": "Snaggy",
        "ItemCategoryId": "458ac9fe-f77c-4cc1-b47f-d5f857413ac1",
        "ItemVersionId": "d3a389fc-29ef-4478-8b3a-2a19632d8260",
        "VisitId": "150962d0-e6b5-4841-b9fd-0cff6e52ae51",
        "DrawingId": null,
        "CreatedAt": "2020-04-30T13:36:56.086857",
        "CreatedBy": "5ce18cc0-d26c-470d-b489-a9ea70df5529",
        "CreatedOn": "00000000-0000-0000-0000-
        0000000000000000000000000000000000000000000000000000000000000000000000000
        0000000000000000000
        "ModifiedAt": "2020-04-30T13:37:16.117427",
        "ModifiedBy": "5ce18cc0-d26c-470d-b489-a9ea70df5529",
        "FrozenAt": null,
        "FrozenBy": null,
        "ModuleInstanceDictionary": {},
        "ItemSectionInstanceDefinitions": [ ]
 }
```

Except for the "ItemSectionInstanceDefinitions" property, the content of this response is the same as in the previous paragraph. To process the response in the RoundItems and Answers, see the previous section.

Note: In the response of this end-point, the Id of a ModuleInstance object refers to the Id of a ModuleInstance. In the end-point response from the previous paragraph, this Id refers to an itemModuleViid.

The ItemSectionInstanceDefinitions array contains the following data per entry (this structure is similar to the structure described in the "3.33.3" section):

```
{
        "Id": "938b424d-0a85-4254-a3e6-f5cf1d41a81e",
        "Index": 1,
        "ItemSectionId": "6a82f9f8-de1b-4abd-8ae3-09e83e5abf41",
        "Name": "Test subject 1",
        "Fields": []
}
```

A Field is a Question in the data model. The Fields array contains the following data per entry:

```
{
        "Id": "bf8bb3ae-22df-4ce4-bd48-e41e396fad31",
        "Index": 1,
        "Name": "Test question",
        "Modules": []
}
```

A Module is a Command in the data model. The Modules array contains the following data per entry:

```
{
        "Id": "499b0e50-accd-4547-974d-1ed890a05a68",
        "ItemModuleTypeId": "7bf16585-70c0-449c-8e28-b6cbef01fb85",
        "ItemModuleId": "51545010-b870-46eb-826a-cf60a8a83ec3",
        "Index": 1,
        "Name": "Test answer",
        "ModuleOptions": []
}
```

Depending on the ItemModuleTypeId, a ModuleOption corresponds to one of the Option tables or the Label table from the data model.

The ModuleOptions array contains the following data per entry:

```
{
        "Id": "89f5fc6e-a4fe-46fd-9a6e-cf816ee2b757",
        "ParentId": null,
        "Index": 6,
        "Name": "For approval"
}
```

3.6.4.4 Response columns

CommandIndex

The CommandIndex column can be determined by searching the itemSectionInstanceDefinition array to the ModuleOptions with the Id value of the ItemModuleId property of the Item.ItemModuleInstance.

QuestionId and QuestionIndex

The QuestionId and QuestionIndex columns are then in the Field object with the Module in the modules array.

Subject and SubjectIndex

The columns SubjectId and SubjectIndex are then back in the entry of the itemSectionInstanceDefinition array of which the Field object is in the Fields array.

3.7 Answer

Central to the data model are the RoundItem and "Answer" tables. These tables contain all the data on the answers given within project rounds to the questions from Snags and Forms. All other tables can be considered as master tables.

Like all IDs, the RondeItem.Id column is unique within the database. This ID is generated when a Snag or Form is linked to a Round or a drawing within a project. So this is not the ID of the snag or the form itself. The Id of the Snag or Form is in the ItemTypeId column. The name of the ItemType is taken from the response of the used end-point and this name is also in the Name column of the ItemType table.

The RondeId column is self-explanatory and refers to the Round table that contains the name of the Round within which the Snag is placed or the answer is filled in.

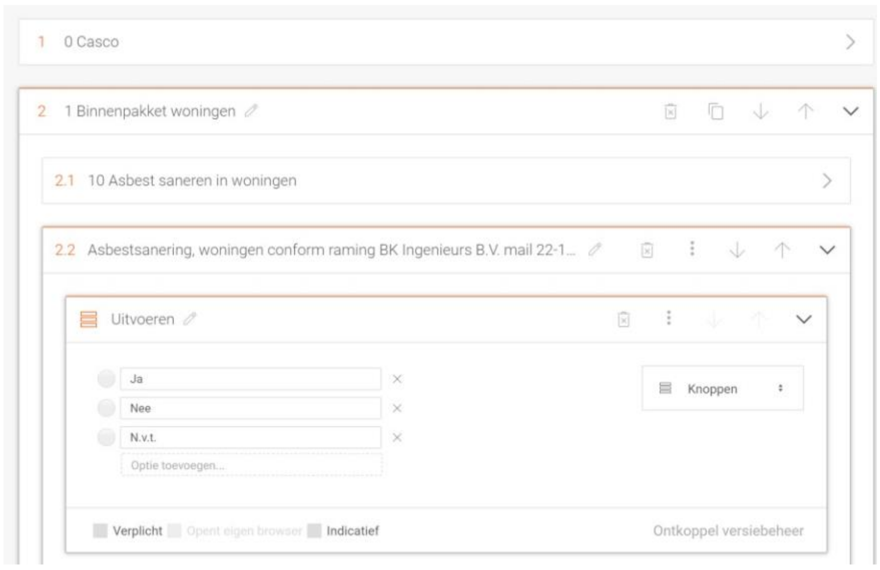
3.7.1 Types of answers

An answer can consist of a free text, a date, a choice of a number of options, choosing employee(s) or taking photos. The columns of the Response table that belong to a particular type of response are:

Answer type	**Affected columns in the response table**
Free text	Answer
Date	Answer
Option choice	The Response column contains the text of the option, and the AnswerId column references a line from the CommandTypeOption table
Employee choice	The response column contains the employee's name and the ReplyId column refers to a line from the User table
Photograph	Number of photos

3.7.1.1 Position of an answer within a snag or form

Each form or snag within Snagstream is organized in a hierarchy of Topics, Questions, and Assignments. The SubjectIndex, QuestionIndex, and CommandIndex columns refer to this hierarchy. The values of these indices associated with the following figure are therefore : SubjectIndex = 2, QuestionIndex = 2 and TaskIndex = 1.



1 0 Casco

2 1 Binnenpakket woningen

2.1 10 Asbest saneren in woningen

2.2 Asbestsanering, woningen conform raming BK Ingenieurs B.V. mail 22-1...

Uitvoeren

Ja

Nee

N.v.t.

Optie toevoegen...

Knoppen

Verplicht Opent eigen browser Indicatief Ontkoppel versiebeheer

3.8 Sample questions

This section contains example SQL statements for answering certain questions from the data model.

Number of Snags linked to a project

```
SELECT count(ri.id)
FROM Round v
JOIN RondeItem ri ON ri. RondeId = v.Id
WHERE v.ProjectId = '2501A64A-2B6F-4B65-B130-DE0244E5DBA7'
AND ri. CategoryId = '458ac9fe-f77c-4cc1-b47f-d5f857413ac1'
```

Number of Snags within a project in which certain types of questions have been answered

```
SELECT count(ri.id)
FROM Round v
JOIN RondeItem ri ON ri. RondeId = v.Id
JOIN Answer a on a.RondeItemId = ri. Id
WHERE v.ProjectId = '2501A64A-2B6F-4B65-B130-DE0244E5DBA7'
AND ri. CategoryId = '458ac9fe-f77c-4cc1-b47f-d5f857413ac1  '
AND a.CommandTypeId IN('837a73eb-f637-4b14-b2eb-ca082a06168f', 'ee2ba447-96b4-44ce-
aae4-
605083a525cf', '4f225325-1e2d-4662-8e6d-fa073bad94a2')
```

In this example, the Ids of the CommandTypes are included:

- Action
- Priority
- Agreement

See also the table in the "CommandTypes" section.

Number of Snags within a project in which a particular employee is linked to a particular question

```
SELECT count(ri.id)
FROM Ronde v
join RondeItem ri ON ri. RondeId = v.Id
join Answer a on a.RondeItemId = ri. Id
WHERE v.ProjectId = '2501A64A-2B6F-4B65-B130-DE0244E5DBA7'
AND ri. CategoryId = '458ac9fe-f77c-4cc1-b47f-d5f857413ac1  '
AND a.CommandTypeId IN ('60619ca6-054d-4e0f-a2f2-e3b78f7a1e89') AND
a.SubjectIndex = 1 AND a.QuestionIndex = 2 AND a.CommandIndex = 3
and a.Answer = 'Employee Name'
```

This example assumes that Topic 1, Question 2, Assignment 3 is a "Assigned to" command.

All snags where under subject with index 1 a Command of the Type Agreement is answered with no

```
SELECT*
FROM RondeItem ri
JOIN Answer a ON a.RondeItemId = ri. Id
JOIN ChosenOption opt ON opt. AnswerId = a.Id
WHERE ri. SubjectIndex = 1
AND ri. CommandTypeId = '4f225325-1e2d-4662-8e6d-fa073bad94a2'
AND ri. Answer = 'No'
```