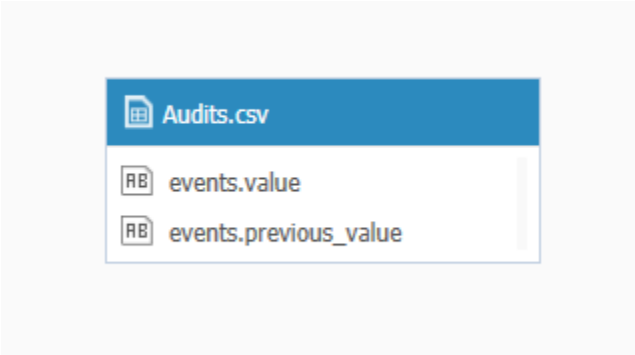
Action (Tables)

The Action (table definition) object represents the tables you import into the ElastiCube Manager. In this object, you define how the tables are to be displayed in the ElastiCube Manager, which columns are included and excluded, and other information such as how to and where to fetch the data from, such as endpoints, query string, headers etc.

For example, the following code sample imports a table called Audits and excludes two columns, events.recipients and events.attachments while transforming the values of columns events.value and events.previous_value to strings.



Audits

```
"Name": "Audits",
      "Public": "True",
      "Schema": "Http",
      "Method": "GET",
      "Base": "[@Settings.Parameters.uri]",
      "Path": "api/v2/tickets/{0}/audits.json",
      "Headers": {
        "Authorization": "Basic
YWxleGV5LmF2cmFtZW5rb0BzaXNlbnNlLmNvbS90b2tlbjpTY09TbzNkR0k0UFJKbldQakZ2dTZxb1lqR3ZEWURRR1dFNnlkYlhG"
      },
                "AccumulativeSettings": {
                "DefaultValue": "20160810",
                "DateTimeFormat": "yyyyMMdd"
            },
      "PathParameters": [ "[@Tables.Tickets.@~Doc.id]" ],
      "QueryParameters": {
                      "api-key": "47fh3hd4n4jc4nnjvjvjfn99nnr",
                      "Begin-date": "[@accumulative]"
                },
      "Body": {},
      "DataFormat": "Json",
      "DataPath": "audits",
      "Transformation": {
        "ExcludeList": [
          "events.recipients",
          "events.attachments"
        ],
        "TransformTypes": [
          {
            "Name": "events.value",
            "Type": "String"
          },
          {
            "Name": "events.previous_value",
            "Type": "String"
          }
        ]
      }
    },
```

It also contains a set of transformation methods that modify how the data is returned in a table.

JSON Format

Name	Type	Mandatory	Comment
Name	String	Yes	The name of a table. This string cannot be longer than 25 characters. **Note**: Underscores in the name are not supported.
Public	Boolean	No	Indicates if the table should be publicly visible. True by default.
Schema	String	Yes	Protocol to use for communicating with a data source endpoint. "Http" or "Https".
Method	String	Yes	HTTP request methods to use. "GET" or "POST".
Base	String	Yes	Data source's REST API endpoint URL.
Path	String	Yes	URI of a resource that the table represents. This value is defined relative to the Base value.
Headers	Object	Yes	String -> String key-value pairs required by the REST service provider. These key-value pairs are typically information you need to provide the RESTful service, but do not want to expose in the URL of the request, for example authentication details.
AccumulativeSettings	Object	No	This object contains keys whose values define the starting point for your accumulative builds. Accumulative Builds are supported only for connectors that support them and for integer and datetime fields. Within the AccumaltiveSettings object, there are two keys that you can define. **DefaultValue**: (Mandatory) Defines the starting point for the build. The first time you must define this value, and for each subsequent build, Sisense updates this value automatically. This can be an integer or datetime value. **DateTimeFormat**:(Optional) When the DefaultValue is a datetime field, this key defines the format for the DefaultValue. Click here for formatting examples. This object should only be defined when using the [@accumulative] InMemory parameter within the same table. For each table that you want to support accumulative builds, you must include the [@accumulative] reference and the AccumulativeSettings object. For more information about InMemory parameters, click here.
PathParameters	Array	Yes	List of strings of path parameters. For more information, click here.
QueryParameters	Object	No	String -> String key-value pairs for specifying query parameters in URL query string.
ArrayQueryParameters	Object	No	String -> array key-value pairs, used for handling arrays in URL query string.

Body	Object	No	HTTP request body.
			
			Sisense supports using nested objects with the Body object for HTTP Post requests.
			
			For example:
			
			"Body": {
			
			"credentials": {
			"username": "MyUser",
			"password": "MyPassword",
			"company": "123456"
			}
			}
			
			Some data sources require parameters represented as JSON objects, such as credentials stored in objects as shown in the example above.

BodyXml	String	No	When the data source expects XML, set the content as the value of BodyXml and leave the Body object empty. As the data source provider expects XML, the value of BodyXml should be in XML format, not JSON. The string should be inside double qoutes " " as displayed in the example below: "BodyXml": "<xml><credentials><username>MyUser</username></credentials></xml>""
DataFormat	String	No	Format of a source data. The only supported value is "Json".

AllowDuplicateElementNames	Boolean	No	Some REST sources return duplicate elements and this may trigger an error. If you receive a Duplicate Name error, set include this value and set it as true to prevent the error and return the full response as expected. Do not include this key:value pair unless you receive the Duplicate Name error as it can decrease performance.
ValidateJsonPropertyName	Boolean	No	Set to true if you receive the error message below. This is caused by unsupported element names according to the JSON standard, such as elements that begin with '.' or '$'. When you set this key to true, Sisense will convert these symbols to underscores, '_' when importing them into Sisense.
Transformation	Transformation	No	Transformation object. For more information, click here.
PagingConfig	PagingConfig	No	Data pagination configuration.
ResponseFormat	String	No	Defines the type of response to be returned from the data source provider to Sisense. Sisense currently accepts JSON (default), CSV, and XML.
Cookies	String	No	When a data source provides cookies in a response to your initial request, this value defines where the cookies are to be sent back to the provider. For example: "Cookies": "[@Credentials.Auth.@~Cookies]", In this example, the data source returns the cookie as the value of Cookies inside the Auth object. This value can then be sent to the data source provider in future requests.
CsvHeader	Boolean	No	When True, returns the Header of content of a CSV.
DataPath	String	Yes	The name of the top level key containing the data set. For more information see DataPath.

Example

Action

```
{
  "Name": "Employees",
  "Public": "True",
  "Schema": "Https",
  "Method": "GET",
  "Base": "https://www.datasource.com",
  "Path": "api/v2/employees.json",
  "Headers": {
    "Authorization": "Basic
YWxleGV5LmF2cmFtZW6zb0BzavNlbnNlLmNdbS90b2tlbjpTY09TbzNkR0k0UFJKbldQakZ2dTZxb1lqR3ZEWURRR1dFNnlkYlhG"
  },
  "PathParameters": [],
  "QueryParameters": {},
  "Body": {},
  "DataFormat": "Json",
  "DataPath": "employees",
  "Transformation": {
    "ExcludeList": [
      "Address",
      "City"
    ],
    "TransformTypes": [
      {
        "Name": "HireDate",
        "Type": "String"
      }
    ]
  }
}
```

DataPath

The framework expects a result set to have the following form:

Example of a possible response

```
{
  "error": {
    "code": 5002,
    "message": "Invalid token.",
    "status": 401,
    "httpMessage": "Unauthorized"
  }
}
```

Usage in the config.json:

Setting DataPath parameter

```
{
        "DataPath":"error",
}
```

Sisense expects the results returned in the format of JSON documents.

If an array is returned, Sisense converts the array into a document with the property "results".

You should reference the "results" as the DataPath in your config.json file, for example:

Example of an array as a response

```
[
  "someElementInArray": {
  }
]
```

You should set:

Setting DataPath parameter

```
{
        "DataPath":"results",
}
```

Transformation

Transformation objects are used to perform changes on a table, which include changing a property, changing a column's type, or even defining a relation with another table.

The follow is a list of Transformation objects you can leverage through the Native REST API:

Name	Purpose
TransformProperties	Change column's name and flatten a nested structure. For more information, see Transform Properties.
TransformTypes	Change column's type. For more information, see TransformTypes.
ExcludeList	Exclude specific columns from a resulting table. For more information, see ExcludeList.
IncludeList	Specify a list of columns that will be shown while hiding everything else. When you transform your data through the TransformProperties object, IncludeList is executed first, then Transform properties. It is important to note the order as IncludeList uses first-level properties and this prevents you from transforming data in deeper levels. For example, if you include the following columns: `"IncludeList": [ "location.latitude", "location.longitude", "location.name", "location.id" ],` then attempt to transform latitude, longitude, name, and id, Sisense will not locate them as only "location" remains.
LinkedTable	Define a relation with another table. For more information, see LinkedTable.

TransformProperties

TransformProperties can be used to give a different name to a column.

JSON Format

Name	Type	Mandatory	Comment
SourceName	String	Yes	Name of the column to be changed.
TargetName	String	Yes	The new name of the column.

Example

TransformProperties

```
{
  "TransformProperties": [{
    "SourceName": "usr_id",
    "TargetName": "id"
  }, {
    "SourceName": "lstnm",
    "TargetName": "Last Name",
  }]
}
```

TransformTypes

The TransformTypes transformer can be used to convert a column's type. For example, if you have a result set with a column "created_at" that contains time stamps in a raw string format, using TransformTypes, you can transform this column to a DateTime type. At runtime the transformer attempts to perform a cast to a target type. In case of failure, the type remains unchanged.

JSON Format

Name	Type	Mandatory	Comment

Name	String	Yes	Name of the column to be changed.
Type	String	Yes	The new column type.

Example

This example was taken from the complete Zendesk example, which you can view here.

TransformProperties

```
{
  "TransformTypes": [{
    "Name": "created_at",
    "Type": "DateTime"
  }, {
    "Name": "updated_at",
        "Type": "String"
  }]
}
```

ExcludeList

The ExcludeList transformer can be used to exclude columns from a table. The transformer is defined as an array of strings.

JSON Format

Array of strings that defines the Column names to be excluded.

Example

TransformProperties

```
{
  "ExcludeList": [
    "changed_at",
    "events.complex_field"
  ]
}
```

LinkedTables

The LinkedTables transformer is used to define a relation with another table.

A result set can contain values that can be used to access other resources. For example, if an endpoint /api/persons.json contains a list of persons and each person entry contains a "person_id", using the person_id, you can query a different endpoint, /api/person/{person_id}.json containing more detailed information.

This essentially produces another logical table Persons_Information. The LinkedTables transformer can be used to define a hidden table Persons_Information like other regular tables and then set a reference to this table inside Persons LinkedTables transformer.

JSON Format

LinkedTables

Array of LinkedTable objects.

LinkedTable

Name	Type	Mandatory	Comment
Name	String	Yes	Linked table name.

LinkedTableReference	String	Yes	Reference to a linked table definition.

Example

LinkedTable

```
{
  "LinkedTables": [{
    "Name": "Comments",
    "LinkedTableReference": "[@Tables.Comments.@~Doc.All]"
  }, {
    "Name": "Audits",
    "LinkedTableReference": "[@Tables.Audits.@~Doc.All]"
  }]
}
```

Pagination

Data sources can contain a lot of data, which means that sending everything in a single large HTTP response may not be efficient or practical. A solution to this problem is to transfer data in fixed size pages. Pagination allows you to modify how large sets of data are split into individual pages.

You can implement pagination when there is more data to load than the data source provides in a single response.

Each data source has its own mechanisms for pagination. In the PagingConfig object, you specify how your connector pages through data sets that you want to import into an ElastiCube.

The Custom REST connector supports the following paging methods.

NextToken: This method passes a token when additional can be returned.

NextURL: This method uses a value that contains the URL for the next set of records.

FirstPage/NextPage: This method defines the first page of the data set and returns the next page based on predefined formulas. This method is useful for files such as CSV where no URLs are provided.

NextPage/In Memory Parameters: This method is similar to the previous method differing in that you must return data sent by the data source after modifying it to return additional pages.

JSON Format

The table below provides a list and descriptions of the relevant keys and values that can be added to the PagingConfig object for paging results.

Name	Type	Mandatory	Comment
PageSize	Integer	Yes	A numeric value indicating the amount of entries per page.
KeepUrl	Boolean	No	Indicates if the next request should use the same URL, e.g. ignoring NextPageURL.
PagingMethod	String	Yes	Name of the method to use. Currently the only supported value is "URL".
NextPageURL	String	Yes	Name of a key in the returned JSON data associated with the URL of the next page.
NextToken	String	No	Name of a key in the returned JSON data associated with a token for fetching the next page. Click here for an example.
Headers	Object	No	String -> String key-value pairs for specifying HTTP request headers.
PathParameters	Array	No	List of strings of path parameters. For more information, see Configuration.
QueryParameters	Object	No	String -> String key-value pairs for specifying query parameters in a URL query string for the next page.

Paging Methods

NextPageUrl

The NextPageUrl method supports pagination by returning a value in the NextPageUrl that defines the URL for the next set of results.

For example, if an endpoint returns a list of 200 person entries per page. The result in JSON contains a URL to the next page.

Example

An endpoint returns a list of 200 person entries per page. The result in JSON contains a URL to the next page.

Example 1

```
{
  "PagingConfig": {
    "PageSize": "200",
    "NextPageURL": "https://www.datasource.com/api/persons/???",
    "PagingMethod": "URL"
  }
}
```

If NextPageURL is undefined then the result set is treated as the final page and no more data is fetched.

NextToken

The NextToken method uses a token that is returned from the data source and referred to in your following requests. When the data source provider does not return a token, this indicates the end of the records.

For example, An endpoint returns a list of 200 person entries per page. The result in JSON contains a token that has to be added to or referenced in the query string to return the next page.

The Source URL is the same. The first request URL is https://www.datasource.com/api/persons.json while the second request URL has to be: https://www.datasource.com/api/persons.json?nextPageToken=QGDDGAETBZ

Example

An endpoint returns a list of 200 person entries per page. The result in JSON contains a token that has to be added to the query string to return the next page. The Source URL is the same. The first request URL is https://www.datasource.com/api/persons.json while the second request URL has to be: https://www.datasource.com/api/persons.json?nextPageToken=QGDDGAETBZ

Pagination

```
// example response
{
  "nextPageToken" : "GIYDAOBNGEYS2MBWKQYDAORQGA5DAMBOGAYDAKZQGAYDALBRGA3TQ===",
  "results" : [{
    ...
  }]
}
```

The following is an example of a table which is using a nextPageToken to page a data set.

Example 2

```
{
      "Name": "Users",
      "Public": "True",
      "Schema": "Https",
      "Method": "GET",
      "Base": "[@Settings.Parameters.uri]",
      "Path": "",
         "Headers": {
                "Authorization": "[@Settings.Parameters.Authorization]"
        },
      "PathParameters": [],
      "QueryParameters": {},
        "PagingConfig": {
                "PagingMethod": "URL",
            "PageSize": "100",
            "KeepUrl": "true",
            "NextToken": "nextPageToken",
            "QueryParameters": {
                 "nextPageToken": "[@Tables.Users.@.nextPageToken]"
            }
          },
          "Body": {},
        "DataPath": "results"
    }
}
```

In the example above, KeepUrl was set to 'true' to continue to use the previous URL. The QueryParameters object is set to take the the nextPageToken to be used when accessing the next set of data. The value of the nextPageToken key is a Sisense token, [@Tables.Users.@.nextPageToken], which refers to the next page token value in a server response document.

When "nextPageToken" is empty from a server response, this indicates a final page and the end of the data. Another indicator that the end of the data was met is an empty response from the service.

FirstPage and NextPage

The third method is to define the first page of the data set and the next page through Sisense In Memory parameters as the values of the FirstPage and NextPage keys.

When returning data where there is no nextPageUrl or nextPageToken provided in the response, such as CSV files, you can calculate dynamically the next page number using a predefined formula.

It is important to have a number of a first page as this page is used to calculate next one.

Example

Example 3

```
{
      "Name": "GeoReport",
      "Public": "True",
      "Schema": "Https",
      "Method": "GET",
      "Base": "[@Settings.Parameters.uri]",
      "Path": "reporting/v1/std/geo",
        "Headers": {},
      "PathParameters": [],
      "QueryParameters": {
            "page_offset": "[@Settings.Parameters.firstPage]"
       },
          "PagingConfig": {
               "PagingMethod": "URL",
            "PageSize": "100",
            "KeepUrl": "true",
            "FirstPage": "[@Settings.Parameters.firstPage]",
              "NextPage": "${[@currentpage] + 1}",
            "QueryParameters": {
                 "page_offset": "[@nextpage]"
            }
          },
          "Body": {},
        "Cookies": "[@Credentials.Auth.@~Cookies]",
        "ResponseFormat":"Csv",
        "CvsHeader":"true",
        "DataPath": ""
    }
}
```

In this example, the value of KeepUrl is “true” to continue to use the previous URL. In the QueryParameters object, you should provide the page token in the next HTTP response.

The FirstPage key is set to a parameter or a direct number that represents the number of a first page. The value of the NextPage is a formula you define that calculates the number of the next page of data to be returned. While you can apply different mathematical operations for NextPage, the formula must be enclosed in the figure brackets with a preceding dollar sign.

In the example above, the [@currentpage] In Memory parameter stores a current page number. Initially, this value is initialized with “FirstPage” value. You should use it in a NextPage formula to calculate a next page. The [@nextpage]In Memory parameter returns a next page number after calculating of a predefined formula.

An empty response from the service indicates an end of the data.

NextPage and In Memory Parameters

The fourth method is similar to the previous method. The difference between the third method and this method is that you must return data sent by the data source after modifying it.

Example

Example

Example 4

```
// example response
{
  "page" : "1",
  "calls" : [{
    ...
  }]
}
```

In the response above, the data source return the current page number. This information needs to be modified, because if you use it to make a next call, you will return the same page, and the reading will never end.

The example below provides a solution to this case:

Example 4

```
{
      "Name": "Calls",
      "Public": "True",
      "Schema": "Https",
      "Method": "GET",
      "Base": "[@Settings.Parameters.uri]",
      "Path": "",
        "Headers": {
            "Authorization": "[@Settings.Parameters.Authorization]"
      },
      "PathParameters": [],
      "QueryParameters": {
            "api-key": "[@Settings.Parameters.apiKey]"
       },
          "PagingConfig": {
                "PagingMethod": "URL",
            "PageSize": "100",
            "KeepUrl": "true",
              "NextPage": "${[@Tables.Calls.@.page] + 1}",
            "QueryParameters": {
                 "page": "[@nextpage]",
                 "api-key": "[@Settings.Parameters.apiKey]"
            }
          },
          "Body": {},
        "DataPath": "calls"
    }
}
```

In this example, KeepUrl is set to “true” to continue to use the previous URL. The QueryParameters key should be provided to use given page token in the next HTTP response.

The NextPage key contains the formula used to calculate a number of a next page. While you can apply different math operations for NextPage, the formula must be enclosed in the figure brackets with a preceding dollar sign. In the example above, the value of NextPage is a token that refers to the page, [@Tables.Calls.@.page] and increases the requested data by one each time.

Accessing Array Items

This functionality is available from Sisense V7.0 and later.

In some cases, you may need to access the first or last items of an array returned by your REST source, for example, if you are returning multiple pages and you want to know the ID of the last item in an array.

The example below shows how you can return the last item of an array:

@Tables.EmailClicks.@.result.emailClick.#lastItem.id

In this example, the ID of the last item in the result.emailClick array will be returned.

JSON Traversing

There are situations when a result set has a nested structure and you want to access some of the nested elements. Transformation objects support "dot notation" naming for traversing, for example:

JSON Traversing

```
{
  "results": [{
    "name": "John",
    "last_name": "Appleseed",
    "locations": {
      "work": {
        "address": "9158 Essex Court Beckley, WV 25801",
        "phone": "+1-202-555-0120"
      },
      "home": {
        "address": "2977 Edgewood Drive Dalton, GA 30721",
        "phone": "+1-202-555-0171"
      }
    }
  }]
}
```

To rename the work phone field into a mobile number, you can use TransformProperties as shown below:

TransformProperties

```
{
  "TransformProperties": [{
    "SourceName": "locations.work.phone",
    "TargetName": "Mobile number"
  }]
}
```