



Client-Side APIs for InterSystems IRIS Business Intelligence

Version 2023.3
2024-05-16

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

1 Using Business Intelligence	1
1.1 Creating a Web Application	1
1.2 Introduction to the Business Intelligence JavaScript API	1
1.2.1 Creating a Business Intelligence Connection	2
1.2.2 Creating and Using a Business Intelligence Data Controller	2
1.3 Introduction to the Business Intelligence REST API	4
1.3.1 Use of Slashes in Cube and KPI Names	5
1.3.2 Notes on the Response Objects	5
DeepSee.js	7
DeepSeeConnector	8
DeepSeeDataController	9
DeepSeeResultSet	12
DeepSeeUtils	15
Business Intelligence REST API	19
/Command/BuildAllRegisteredGroups	20
/Command/BuildCube	22
/Command/BuildOneRegisteredGroup	23
/Command/BuildRegistryMap	24
/Command/GetActiveRegistry	26
/Command/GetCubeSize	27
/Command/GetLastUpdate	28
/Command/getSynchScheduleParameters	29
/Command/IsValidCubeSchedule	30
/Command/IsValidGroup	31
/Command/RepairBuild	35
/Command/SetActiveRegistry	36
/Command/SynchronizeCube	37
/Command/WriteToRegistry	38
/Data/GetDSTIME	39
/Data/KPIExecute	40
/Data/MDXCancelQuery/:queryID	42
/Data/MDXDrillthrough	43
/Data/MDXExecute	45
/Data/MDXUpdateResults/:queryID	48
/Data/PivotExecute	49
/Data/TestConnection	51
/Info/Cubes	52
/Info/Dashboards	54
/Info/FilterMembers/:datasource/:filterSpec	55
/Info/Filters/:datasource	57
/Info/KPIs	59
/Info/ListingFields/:cube	60
/Info/Listings/:cube	61
/Info/Measures/:cube	63
/Info/NamedFilters/:cube	64
/Info/Pivots	65

/Info/PivotVariableDetails/:cube/:variable	66
/Info/PivotVariables/:cube	67
/Info/QualityMeasures/:cube	68
/Info/TestConnection	69

1

Using Business Intelligence

This page introduces the JavaScript and REST APIs for InterSystems IRIS® data platform [Business Intelligence](#). These APIs let you execute [MDX queries](#) and retrieve information about Business Intelligence [model elements](#).

1.1 Creating a Web Application

In any scenario (whether you use the JavaScript API or you use the REST services directly), a web application is responsible for handling the requests. You can use the system-defined web application (`/api/deepsee`) or you can create and use a different web application. The requirements for this web application are as follows:

- You must place your client file or files within the directory structure served by this web application.
- You must specify the **Dispatch Class** option, which specifies how this web application handles REST requests. For Business Intelligence REST requests, use one of the following:
 - `%Api.DeepSee` — Use this class if your client application must be able to connect to different namespaces. In this case, when you connect to an InterSystems IRIS server, you must specify the namespace to use.
The system-defined web application (`/api/deepsee`) uses this dispatch class.
 - `%DeepSee.REST.v2` — Use this class if the REST requests should be tied to a specific namespace (the namespace for the web application).

The class `%DeepSee.REST.v1` is also available for use, but is documented only within the class reference.

1.2 Introduction to the Business Intelligence JavaScript API

The Business Intelligence JavaScript API is provided by the file `DeepSee.js`, which is in the `install-dir/CSP/broker` directory. This JavaScript library enables you to interact with Business Intelligence from a client that is based on JavaScript. The functions in this library are a wrapper for a REST-based API for Business Intelligence. (You can also use the [REST API](#) directly.)

To use this library:

1. Create a web application as described in the [previous section](#).

Or use the web application `/api/deepsee`, which is provided as part of the installation.

2. In your JavaScript client code:

- a. Include the files `DeepSee.js` and `zenCSLM.js`.
- b. Create a Business Intelligence connection object. This contains information needed to connect to an InterSystems IRIS server.
- c. Create a Business Intelligence data controller object that uses the connection object.

The data controller object enables you to interact with a Business Intelligence data source, which you specify either via an MDX query or via the name of a pivot table.

- d. Use the **`runQuery()`** method of the data controller. If the data source is an MDX query, Business Intelligence executes that query. If the data source is a pivot table, Business Intelligence executes the query defined by the pivot table.
- e. Invoke other methods of the data controller object to examine the query results, to drill down or drill through, and so on.

The following subsections give the details.

The library `DeepSee.js` also provides utility functions that provide information about Business Intelligence model elements. Use these to obtain lists of available cubes, available measures in a cube, and so on.

1.2.1 Creating a Business Intelligence Connection

To create a Business Intelligence connection object, use code like the following:

```
connection = new DeepSeeConnection(username,password,host,application,namespace);
```

Where:

- *username* is an InterSystems IRIS username that can access the given host.
- *password* is the associated password.
- *host* is the server name for the machine on which InterSystems IRIS is running.
- *application* is the name of the web application.
- *namespace* is the name of the namespace to access. If the web application is tied to a namespace, this argument is not needed.

1.2.2 Creating and Using a Business Intelligence Data Controller

The data controller object enables you to interact with Business Intelligence data sources. The primary interaction is as follows:

- In a suitable part of the page logic (such as when the page is loaded or when a button is pressed), create a Business Intelligence data controller and execute a query.

When you create a data controller, you specify one or two callback functions to be run when data is available; *finalCallback* is required, but *pendingCallback* is optional.

- When pending results are available, Business Intelligence calls the method specified by *pendingCallback*, if specified.

This method, which you write, uses the results that are available in the data controller object. The method typically draws page contents.

- When the query has completed, Business Intelligence calls the method specified by *finalCallback*.

This method, which you write, uses the results that are available in the data controller object. The method typically draws page contents.

Any method that executes a query uses the system described here; see the subsections for details and examples. Other methods return data synchronously.

1.2.2.1 Creating a Business Intelligence Data Controller and Executing a Query

In a suitable part of the client code (such as within the page initialization logic), do the following:

1. Create a configuration object that has the following properties:
 - *connection* — Specifies the name of a Business Intelligence data connector object; see the [previous section](#).
 - *widget* — Specifies the id of the HTML element on the page that will use the data controller
 - *type* — Specifies the type of data source; use either 'MDX' or 'PIVOT'
 - *initialMDX* — Specifies an MDX SELECT query; use this if type is 'MDX'
 - *pivotName* — Specifies the logical name of a pivot table; use this if type is 'PIVOT'
 - *showTotals* — Specifies whether to display totals. Specify either `true` or `false`
2. Create a data controller object with code like the following:

```
var dc = new DeepSeeDataController(configuration,finalCallback,pendingCallback);
```

Where *configuration* is the configuration object from the previous step, *finalCallback* is the name of a callback function on this page, and *pendingCallback* is the name of another callback function on this page. *finalCallback* is required, but *pendingCallback* is optional.

3. Call the **runQuery()** method of the data controller. Or run some other method that executes a query, such as **runDrillDown()** or **runListing()**.

For example:

```
function initializePage() {
  ...
  configuration.connection = new DeepSeeConnection(username,password,host,application,namespace);
  dc = new DeepSeeDataController(configuration,drawChart);
  dc.runQuery();
}
```

1.2.2.2 Using Data Returned by the Data Controller

The page must also implement the callback function or functions referred to in the previous step. These callbacks should update the page as needed, using data obtained from the data controller object.

In each case, the [data controller object](#) is passed to the function as the argument.

The following shows a partial example:

```
function drawChart(dataController) {  
    var resultSet = dataController.getCurrentData();  
    ...  
    var chartDataPoint;  
    var chartLabel;  
    var chartData = [];  
  
    for (var i = 1; i <= resultSet.getRowCount(); ++i) {  
        for (var j = 1; j <= resultSet.getColumnCount(); ++j) {  
            chartDataPoint = resultSet.getOrdinalValue(i,j);  
            chartLabel = resultSet.getOrdinalLabel(2,i);  
            chartData[chartData.length] = { "country":chartLabel[0],"revenue":chartDataPoint};  
        }  
    }  
    ...  
}
```

The `getCurrentData()` method of the data controller returns another object, the result set object. That object provides methods for examining the results of the query. The example here shows some of them.

1.3 Introduction to the Business Intelligence REST API

Internally, the [JavaScript API](#) uses the Business Intelligence REST API, which you can also use directly, as follows:

1. Create a [web application](#).

Or use the web application `/api/deepsee`, which is provided as part of the installation.

2. In your JavaScript client code, create and send HTTP requests to the desired target REST services.

If your web application uses the dispatch class `%Api.DeepSee`, the target URL must include the Business Intelligence REST API version number and the target namespace name as part of the application path. It should take the following form:

```
[protocol]://[baseURL]/[appName]/[version]/[namespace]/[APICall]
```

where `[baseURL]` refers to your instance, `[appName]` is the path which is defined as the web application's name, `[version]` is the version of the REST API which you want to invoke, `[namespace]` is the target namespace, and `[APICall]` is the actual rest call (for example, `/Info/Cubes`). For example:

```
https://data.mycompany.com/api/deepsee/v2/sales/Info/Cubes
```

If your web application uses the dispatch class `%DeepSee.REST.v2`, the REST API version number and the target namespace are implicit to the web application definition and can therefore be omitted from the target URL. In other words, the target URL should take the following form:

```
[protocol]://[baseURL]/[appName]/[APICall]
```

For example:

```
https://data.mycompany.com/mycustombiapp/Info/Cubes
```

Note: The client must accept JSON. The Accept header of the request must either specify `application/json` or not declare a format.

3. Examine the response objects and use as applicable.

1.3.1 Use of Slashes in Cube and KPI Names

It is relatively common to use slashes (/) in the logical names of cubes and other items, because the slash character is the token that separates a folder name from a short item name. For example, a cube might have the logical name `RelatedCubes/Patients`

You *can* directly use these logical names unmodified in URL parameters (as well as in the request bodies). The applicable Business Intelligence REST services account for logical names that include slashes. The logic, however, requires you to follow a naming convention (depending on which REST services you plan to use). Specifically, do not have an item with a logical name that is the same as the name of a folder used by another logical name. For example, if you have an item called `mycubes/test/test1`, you should not have an item called `mycubes/test`.

The reason for this restriction is that when you use a REST service that uses another argument after the logical name, part of the name is interpreted as another argument if the first part of the name matches an existing item. Consider the following REST call:

```
https://localhost/api/deepsee/v2/Info/FilterMembers/:mycubename/:filterspec
```

Here *mycubename* is the logical name of a cube and *filterspec* is the specification for a filter provided by that cube. Now consider this REST call with `mycubes/test/test1` as the name of the cube:

```
https://localhost/api/deepsee/v2/Info/FilterMembers/:mycubes/test/test1/:filterspec
```

In order to interpret the slash characters, the system first attempts to find a cube named `mycubes` and then attempts to find a cube named `mycubes/test`, and so on. When the system finds the first item that matches the apparent name, the REST call uses that item, and the remainder of the string is interpreted as the next argument.

1.3.2 Notes on the Response Objects

For most of the REST calls, the response objects contain the property `Info`, which contains information about the request and response. This object contains the property `Error`, which equals one of the following:

- Null — This indicates that no error occurred.
- An object that contains the properties `ErrorCode` and `ErrorMessage` — This object contains details about the error that you can use to determine whether and how to proceed.

If no error occurred, the response object also contains the property `Result`, which is an object containing the requested values.

In general, your client code should first check the `Info.Error` property and then determine how to proceed.

For example, a response object might look like this (with white space added for readability):

```
{ "Info":
  { "Error":
    { "ErrorCode": "5001",
      "ErrorMessage": "ERROR #5001: Cannot find Subject Area: 'SampleCube'" }
  }
}
```

In contrast, if no error occurred, the `Info.Error` property is null and the `Result` contains the result that you requested. For example:

```
{ "Info":
  { "Error": "",
    "BaseCube": "DemoMDX",
    "SkipCalculated": 0 },
  "Result":
    { "Measures":
      [
        { "name": "%COUNT", "caption": "%COUNT", "type": "integer", "hidden": 0, "factName": "" },
        { "name": "Age", "caption": "Age", "type": "integer", "hidden": 0, "factName": "MxAge" }
      ]
    }
}
```

DeepSee.js

This reference section provides information on the JavaScript API for InterSystems IRIS Business Intelligence. This API is provided by the file DeepSee.js.

DeepSeeConnector

Enables you to connect to Business Intelligence data sources.

Where This Object Is Available

This object is available in client-side JavaScript code, if that code includes DeepSee.js. See [Introduction to DeepSee.js](#).

Creating This Object

To create a Business Intelligence data connector object, use code like the following:

```
var connection = new DeepSeeConnection(username,password,host,application,namespace);
```

Where:

- *username* is an InterSystems IRIS® data platform username that can access the given host.
- *password* is the associated password.
- *host* is the server name for the machine on which InterSystems IRIS is running.
- *application* is the name of the web application.
- *namespace* is the name of the namespace to access.

Properties of This Object

This object provides the following properties:

- *username* is an InterSystems IRIS username that can access the given host.
- *password* is the associated password.
- *path* is the base URL for the web services.

Methods of This Object

A data connector object does not provide any methods.

DeepSeeDataController

Enables you to work with a Business Intelligence data source.

Where This Object Is Available

This object is available in client-side JavaScript code, if that code includes `DeepSee.js`. See [Introduction to DeepSee.js](#).

Creating This Object

To create a Business Intelligence data controller object, use code like the following:

```
var controller = new DeepSeeDataController(configuration,finalCallback,pendingCallback);
```

Where:

- *configuration* is an object that has the following properties:
 - *connection* — Is a Business Intelligence [data connector object](#).
 - *widget* — Specifies the id of the HTML element on the page that will use the data controller
 - *type* — Specifies the type of data source; use either 'MDX' or 'PIVOT' (case-sensitive)
 - *initialMDX* — Specifies an MDX SELECT query; specify this property if *type* is 'MDX'
 - *pivotName* — Specifies the logical name of a pivot table; specify this property if *type* is 'PIVOT'
 - *showTotals* — Specifies whether to display totals; specify this property as `true` or `false`
- *finalCallback* is the name of a callback function on this page
For `runQuery()` and other methods of this object, when the query is completed, the system invokes this function.
- *pendingCallback* (optional) is the name of another callback function on this page.
For `runQuery()` and other methods of this object, when pending results are available, the system invokes this function, if this argument is supplied.

Methods of This Object

The Business Intelligence data controller object provides the following JavaScript methods:

applyFilter()

```
applyFilter(filterInfo)
```

Where *filterInfo* is an object that contains the *filterName* and *filterSpec* properties.

This method adds the given filter to the filters used by the data controller, reruns the query, and then invokes the [callback functions](#) associated with the data controller object.

This method has no return value.

attachTotals()

```
attachTotals(rowTotals,columnTotals,reattach)
```

Where:

- *rowTotals* is `true` or `false`, depending on whether you want to attach totals for the rows.

- *columnTotals* is `true` or `false`, depending on whether you want to attach totals for the columns.
- *reattach* is `true` or `false`, depending on whether you want to update the most recently saved state.

If *reattach* is `true`, the system updates the most recently saved state. If *reattach* is `false`, the system adds a new state object to the stack.

This method attaches totals to the data controller object and then invokes the [callback functions](#) associated with the data controller object.

This method has no return value.

getCurrentData()

```
getCurrentData()
```

Returns a [Business Intelligence result set object](#) that contains the results from the query currently defined by the data controller object. See the reference for the [DeepSeeResultSet](#) object.

getCurrentQueryText()

```
getCurrentQueryText()
```

Returns the text of the query currently defined by the data controller object.

runDrillDown()

```
runDrillDown(axis, position)
```

Where:

- *axis* is the number of the axis on which you want to perform the drilldown action. Specify 1 for column or 2 for rows.
- *position* is the position (1-based) on that axis where you want to perform the drilldown action.

This method executes the given drilldown action, and then invokes the [callback functions](#) associated with the data controller object.

This method has no return value.

runListing()

```
runListing(startRow, startCol, endRow, endCol, listingName)
```

Where:

- *startRow* and *startCol* are the first row and column number of the results for which you want a detail listing. Specify 1 for the first row or column.
- *endRow* and *endCol* are the last row and column number for which you want a detail listing.
- *listingName* is the logical name of a detail listing.

This method executes the given detail listing for one or more cells of the results, and then invokes the [callback functions](#) associated with the data controller object.

This method has no return value.

runQuery()

```
runQuery()
```

Executes the query currently defined by the data controller object. When the query is pending or completed, the system invokes the [callback functions](#) associated with the data controller object.

This method has no return value.

sortResults()

```
sortResults(axis, position, direction, sortType)
```

Where:

- *axis* is the number of the axis you want to sort. Specify 1 for column or 2 for rows.
- *position* is the position (1-based) on that axis where you want to sort.
For example, to sort by the third column, specify *axis* as 1 and *position* as 3.
- *direction* is the direction in which to sort. Specify 1 for ascending sort or -1 or descending sort.
- *sortType* specifies how to sort. If this is '' or 'numeric' (case-insensitive), numeric sorting is used. Otherwise, string sorting is used.

This method sorts the results as requested and then invokes the [callback functions](#) associated with the data controller object.

This method has no return value.

undoLastAction()

```
undoLastAction()
```

Undoes the last change and invokes the [callback functions](#) associated with the data controller object.

This method has no return value.

DeepSeeResultSet

Enables you to work with the results of a Business Intelligence query.

Where This Object Is Available

This object is available in client-side JavaScript code, if that code includes DeepSee.js. See [Introduction to DeepSee.js](#).

Creating This Object

To create a result set, call the **getCurrentData()** method of the [data controller object](#).

Or use code like the following:

```
var resultset = new DeepSeeResultSet(connection, widget, wait, timeout);
```

Where:

- *connection* — Is a Business Intelligence [data connector object](#).
- *widget* — Specifies the id of the HTML element on the page that will use the data controller.
- *wait* — Boolean. If this parameter is false, the server should respond immediately. If this parameter is true, the server should not respond until the query has completed or timed out.
- *timeout* — Specifies how long (in seconds) the server should wait before returning final but incomplete query results.

Properties of This Object

The Business Intelligence result set object provides the following properties:

- *connection* — Is a Business Intelligence [data connector object](#).
- *widget* — Specifies the id of the HTML element on the page that will use the data controller.
- *wait* — Boolean. If this property is false, the server should respond immediately. If this property is true, the server should not respond until the query has completed or timed out.
- *timeout* — Specifies how long (in seconds) the server should wait before returning final but incomplete query results.
- *data* — Contains the query results.
- *pollInterval* — Specifies how long (in milliseconds) to wait before asking the server for updates to pending results. The default is 1000.

Methods of This Object

The Business Intelligence result set object provides the following JavaScript methods:

getColumnCount()

```
getColumnCount()
```

Returns the number of columns in the result set.

getCubeName()

```
getCubeName()
```

Returns the name of the cube currently in use.

getErrorMessage()

```
getErrorMessage()
```

This method returns the text of the error message contained in the response object, if any.

getOrdinalLabel()

```
getOrdinalLabel(axis, position)
```

Where:

- *axis* is the number of the axis whose labels you want to obtain. Specify 1 for column or 2 for rows.
- *position* is the position on that axis whose labels you want to obtain. Specify 1 for the first position on the axis.

Returns an array of strings, corresponding to the labels at the given position on the given axis.

getOrdinalValue()

```
getOrdinalValue(rowNo,colNo,formatted)
```

Where:

- *rowNo* is the row number (1-based)
- *colNo* is the column number (1-based)

Returns the value in the given cell.

getQueryStatus()

```
getQueryStatus()
```

Returns a numeric value indicating whether the query has completed. This number is 100 if the query has completed. Otherwise, this number is less than 100.

getRowCount()

```
getRowCount()
```

Returns the number of rows in the result set.

isError()

```
isError()
```

This method returns `true` if the response object indicates that an error occurred. Returns `false` otherwise.

runMDXQuery()

```
runMDXQuery(mdx,finalCallback,pendingCallback,filters)
```

Where:

- *mdx* is an MDX SELECT query.
- *finalCallback* is the name of a callback function on this page.

- *pendingCallback* (optional) is the name of another callback function on this page.
- *filters* specifies additional filters to apply to the query.

This method executes the given MDX query. When the query is pending or completed, the system invokes the given callback functions.

This method has no return value.

runMDXDrillQuery()

```
runMDXDrillQuery(mdx,finalCallback,pendingCallback,listing,fieldList,filters)
```

Where:

- *mdx* is an MDX SELECT query.
- *finalCallback* is the name of a callback function on this page.
- *pendingCallback* (optional) is the name of another callback function on this page.
- *listing* is the name of a detail listing.
- *fieldList* specifies a list of listing fields. Specify either *listing* or *fieldList*.
- *filters* specifies additional filters to apply to the query.

This method executes the given drillthrough query. When the query is pending or completed, the system invokes the given callback functions.

This method has no return value.

runPivot()

```
runPivot(pivot,finalCallback,pendingCallback,filters)
```

Where:

- *pivot* is the logical name of pivot table.
- *finalCallback* is the name of a callback function on this page.
- *pendingCallback* (optional) is the name of another callback function on this page.
- *filters* specifies additional filters to apply to the query.

This method executes the MDX query defined by the given pivot table. When the query is pending or completed, the system invokes the given callback functions.

This method has no return value.

DeepSeeUtils

Provides additional methods for working with Business Intelligence.

Where This Object Is Available

This object is available in client-side JavaScript code, if that code includes DeepSee.js. See [Introduction to DeepSee.js](#).

Creating This Object

When you include DeepSee.js in your client code, the *DeepSeeUtils* object is automatically available.

Methods of This Object

The *DeepSeeUtils* object provides the following methods:

getCubeList()

```
getCubeList(connection, finalFunc)
```

Where:

- *connection* — Is a Business Intelligence [data connector object](#).
- *finalFunc* — Specifies the name of the function to execute when results are available

This method retrieves information about the available cubes. It calls the [POST /Info/Cubes](#) REST call. The response object from that call is available to the function specified by *finalFunc*.

getCubeListings()

```
getCubeListings(connection, finalFunc)
```

Where:

- *connection* — Is a Business Intelligence [data connector object](#).
- *cubename* — Specifies the logical name of a cube.
- *finalFunc* — Specifies the name of the function to execute when results are available

This method retrieves information about the listings available to the given cube. It calls the [POST /Info/Listings/:cube](#) REST call. The response object from that call is available to the function specified by *finalFunc*.

getDashboardList()

```
getDashboardList(connection, finalFunc)
```

This method retrieves information about the available dashboards. It calls the [POST /Info/Dashboards](#) REST call. The response object from that call is available to the function specified by *finalFunc*.

getErrorMessage()

```
getErrorMessage(data)
```

Where *data* is a response object returned by any of the Business Intelligence [REST services](#).

This method returns the text of the error message contained in that object, if any.

getFiltersForDataSource()

```
getFiltersForDataSource(connection, cubename, finalcallback)
```

Where:

- *connection* — Is a Business Intelligence [data connector object](#).
- *cubename* — Specifies the logical name of a cube.
- *finalcallback* — Specifies the name of the function to execute when results are available

This method retrieves information about the available filters for the given cube. It calls the [POST /Info/Filters/:data-source](#) REST call. The response object from that call is available to the function specified by *finalFunc*.

getMembersForFilter()

```
getMembersForFilter(connection, cubeName, filterSpec, finalCallback)
```

Where:

- *connection* — Is a Business Intelligence [data connector object](#).
- *cubename* — Specifies the logical name of a cube.
- *filterSpec* — Specifies the filter whose members you want to retrieve.
- *finalcallback* — Specifies the name of the function to execute when results are available

This method retrieves information about members of the given filter. It calls the [POST /Info/FilterMembers/:data-source/:filterSpec](#) REST call. The response object from that call is available to the function specified by *finalFunc*.

getPivotList()

```
getPivotList(connection, finalFunc)
```

Where:

- *connection* — Is a Business Intelligence [data connector object](#).
- *finalFunc* — Specifies the name of the function to execute when results are available

This method retrieves information about the available pivot tables. It calls the [POST /Info/Pivots](#) REST call. The response object from that call is available to the function specified by *finalFunc*.

getResultsAsArray()

```
getResultsAsArray(data)
```

Where *data* is a response object returned by any of the Business Intelligence [REST services](#).

This method returns an array of results from that object as follows:

1. If the response object contains an `Result.Filters` array, the method returns that array.
2. Otherwise, if the response object contains an `Result.FilterMembers` array, the method returns that array.
3. Otherwise, if the response object contains an `Result.Listings` array, the method returns that array.

Otherwise the method returns nothing.

isError()

```
isError(data)
```

Where *data* is a response object returned by any of the Business Intelligence [REST services](#).

This method returns `true` if the response object indicates that an error occurred. Returns `false` otherwise.

Business Intelligence REST API

This reference section provides information on the REST services for InterSystems IRIS® data platform [Business Intelligence](#). These services are provided by the class %DeepSee.REST.v2. See [Using the Business Intelligence REST API](#).

/Command/BuildAllRegisteredGroups

Builds all cubes registered in the current cube registry.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint is called with an empty body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/BuildAllRegisteredGroups`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

`{}`

Example Response

```
{
  "buildStats": {
    "buildStats_1": {
      "caption": "Filter RDoctors",
      "elapsedTime": 0.01413,
      "errors": 0,
      "event": "build",
      "eventId": "124",
      "expressionTime": 0.000039,
      "factCount": 40,
      "iKnowTime": 0,
      "missingReferences": 0,
      "status": 1
    },
    "buildStats_2": {
      "caption": "Filter RPatients",
      "elapsedTime": 0.148839,
      "errors": 0,
      "event": "build",
      "eventId": "126",
      "expressionTime": 0.053942,
      "factCount": 1000,
      "iKnowTime": 0,
      "missingReferences": 0,
      "status": 1
    },
    "buildStats_3": {
      "caption": "Patients",
      "elapsedTime": 0.490752,
      "errors": 0,
      "event": "build",
      "eventId": "122",
      "expressionTime": 0.343703,
      "factCount": 1000,
      "iKnowTime": 0,
      "missingReferences": 0,
      "status": 1
    }
  }
}
```



```

        "missingReferences": 0,
        "status": 1
    },
    "buildStats_4": {
        "caption": "RCities",
        "elapsedTime": 0.010991,
        "errors": 0,
        "event": "build",
        "eventId": "123",
        "expressionTime": 0.000003,
        "factCount": 9,
        "iKnowTime": 0,
        "missingReferences": 0,
        "status": 1
    },
    "buildStats_5": {
        "caption": "RCityRainfall",
        "elapsedTime": 0.006289,
        "errors": 0,
        "event": "build",
        "eventId": "127",
        "expressionTime": 0,
        "factCount": 0,
        "iKnowTime": 0,
        "missingReferences": 0,
        "status": 1
    },
    "buildStats_6": {
        "caption": "RDoctors",
        "elapsedTime": 0.014014,
        "errors": 0,
        "event": "build",
        "eventId": "125",
        "expressionTime": 0.000041,
        "factCount": 40,
        "iKnowTime": 0,
        "missingReferences": 0,
        "status": 1
    },
    "buildStats_7": {
        "caption": "RPatients",
        "elapsedTime": 0.136887,
        "errors": 0,
        "event": "build",
        "eventId": "128",
        "expressionTime": 0.051799,
        "factCount": 1000,
        "iKnowTime": 0,
        "missingReferences": 0,
        "status": 1
    },
    "buildStats_8": {
        "caption": "RPatientsDependsOn",
        "elapsedTime": 0.146196,
        "errors": 0,
        "event": "build",
        "eventId": "129",
        "expressionTime": 0.051907,
        "factCount": 1000,
        "iKnowTime": 0,
        "missingReferences": 0,
        "status": 1
    },
    "buildStats_9": {
        "caption": "RPatientsInherit",
        "elapsedTime": 0.163897,
        "errors": 0,
        "event": "build",
        "eventId": "130",
        "expressionTime": 0.059377,
        "factCount": 1000,
        "iKnowTime": 0,
        "missingReferences": 0,
        "status": 1
    }
},
"Error": "",
"status": 1
}

```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/BuildCube

Builds the target cube, and records and reports build event information.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

cubeName

Required. The name of the cube to be built.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/BuildCube`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "cubeName": "Patients"
}
```

Example Response

```
{
  "buildStats": {
    "caption": "Patients",
    "elapsedTime": 0.594205,
    "errors": 0,
    "event": "build",
    "eventId": "119",
    "expressionTime": 0.396416,
    "factCount": 1000,
    "iKnowTime": 0,
    "missingReferences": 0,
    "status": 1
  },
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/BuildOneRegisteredGroup

Builds all cubes in a registered cube group in the order in which they are listed and returns statistics on the build process.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

groupName

Required. The name of the cube group to be built.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/BuildOneRegisteredGroup`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "groupName": "Group 1"
}
```

Example Response

```
{
  "buildStats": {
    "caption": "Patients",
    "elapsedTime": 0.594205,
    "errors": 0,
    "event": "build",
    "eventId": "119",
    "expressionTime": 0.396416,
    "factCount": 1000,
    "iKnowTime": 0,
    "missingReferences": 0,
    "status": 1
  },
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/BuildRegistryMap

Builds a map of the current state of the Cube Registry and all unregistered cubes on the system.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint is called with an empty body.

Example Request

- Request Method:*

POST

- Request URL:*

`https://localhost/api/deepsee/v2/Command/BuildRegistryMap`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- Request Body:*

```
{}
```

Example Response

```
{
  "registryMap":
  {
    "is a sample Cube Registry for QD testing\\"",
    "lookup": {
      "cubes": [
        {
          "FILTERRDOCTORS": [
            2,
            2
          ],
          "FILTERRPATIENTS": [
            2,
            4
          ],
          "PATIENTS": [
            1,
            1
          ],
          "RCITIES": [
            2,
            1
          ],
          "RCITYRAINFALL": [
            2,
            5
          ]
        }
      ]
    }
  }
}
```

```

        "RDOCTORS": [
            2,
            3
        ],
        "RPATIENTS": [
            2,
            6
        ],
        "RPATIENTSDEPENDSON": [
            2,
            7
        ],
        "RPATIENTSINHERIT": [
            2,
            8
        ]
    },
    "groups": [
        {
            "Group 1": [
                1
            ],
            "Group 2": [
                2
            ]
        }
    ],
    "Error": "",
    "status": 1
}

```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/GetActiveRegistry

Builds all cubes in a registered cube group in the order in which they are listed and returns statistics on the build process.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint is called with an empty body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/GetActiveRegistry`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

`{}`

Example Response

```
{
  "activeRegistry": "DeepSee.CubeManager.CubeRegistryDefinition",
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/GetCubeSize

Given the name of a cube, returns the number of rows in its fact table.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

cubeKey

Required. The logical name of the cube whose size is in question.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/GetCubeSize`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "cubeKey": "Patients"
}
```

Example Response

```
{
  "cubeSize": 1000,
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/GetLastUpdate

Retrieves the timestamp of the last registered data update for the cube. Updates performed outside of the Cube Manager will not be recognized in this check.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

cubeKey

Required. The logical name of the cube for which to retrieve the last update.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/GetLastUpdate`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "groupName": "Group 1"
}
```

Example Response

```
{
  "tSC": 1,
  "lastUpdate": "2020-04-30 16:58:00",
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/getSynchScheduleParameters

A convenience method that retrieves and displays the parameters of %SYS.Task.DeepSeeSynchronize.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint is called with an empty body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/getSynchScheduleParameters`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{}
```

Example Response

```
{
  "synchParams": {
    "dailyFrequency": 1,
    "dailyFrequencyTime": 0,
    "dailyIncrement": 32,
    "dailyStartTime": 0
  },
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/IsValidCubeSchedule

Performs validation on current cube settings.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

cube

Required. The name of the cube the schedule of which to validate.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/IsValidCubeSchedule`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "cube": "HoleFoods Base"
}
```

Example Response

```
{
  "isValid": 1,
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).


```
],
"4": [
  {
    "JMD1340": [
      1
    ]
  }
],
"5": [
  {
    "MYCUBE89212": [
      1
    ]
  }
],
"6": [
  {
    "NEWHOLEFOODS": [
      1
    ]
  }
],
"7": [
  {
    "PATIENTS": [
      1
    ]
  }
],
"8": [
  {
    "PREBUILTHOLEFOODS": [
      1
    ]
  }
],
"9": [
  {
    "FILTERRDOCTORS": [
      2
    ],
    "FILTERRPATIENTS": [
      5
    ],
    "RCITIES": [
      1
    ],
    "RCITYRAINFALL": [
      3
    ],
    "RDOCTORS": [
      4
    ],
    "RPATIENTS": [
      6
    ],
    "RPATIENTSDEPENDSON": [
      7
    ],
    "RPATIENTSINHERIT": [
      8
    ]
  }
],
},
"cubes": {
  "DEPENDSONHOLEFOODS": [
    1
  ],
  "EXPRESSIONHOLEFOODS": [
    2
  ],
  "FILTERRDOCTORS": [
    9,
    {
      "dependents": [
        2,
        {
          "FILTERRPATIENTS": [],
          "RPATIENTS": []
        }
      ]
    }
  ],
  "FILTERRPATIENTS": [
```

```

    9
  ],
  "HOLEFOODS": [
    3
  ],
  "JMD1340": [
    4
  ],
  "MYCUBE89212": [
    5
  ],
  "NEWHOLEFOODS": [
    6
  ],
  "PATIENTS": [
    7
  ],
  "PREBUILTHOLEFOODS": [
    8
  ],
  "RCITIES": [
    9,
    {
      "dependents": [
        7,
        {
          "FILTERRDOCTORS": [],
          "FILTERRPATIENTS": [],
          "RCITYRAINFALL": [],
          "RDOCTORS": [],
          "RPATIENTS": [],
          "RPATIENTSDEPENDSON": [],
          "RPATIENTSINHERIT": []
        }
      ]
    }
  ],
  "RCITYRAINFALL": [
    9
  ],
  "RDOCTORS": [
    9,
    {
      "dependents": [
        3,
        {
          "RPATIENTS": [],
          "RPATIENTSDEPENDSON": [],
          "RPATIENTSINHERIT": []
        }
      ]
    }
  ],
  "RPATIENTS": [
    9
  ],
  "RPATIENTSDEPENDSON": [
    9
  ],
  "RPATIENTSINHERIT": [
    9
  ],
  },
  "buildOrders": {
    "1": [
      1,
      {
        "1": [
          "DEPENDSONHOLEFOODS"
        ]
      }
    ],
    "2": [
      1,
      {
        "1": [
          "EXPRESSIONHOLEFOODS"
        ]
      }
    ],
    "3": [
      1,
      {
        "1": [
          "HOLEFOODS"
        ]
      }
    ]
  }
}

```

```
    ]
  },
  "4": [
    1,
    {
      "1": [
        "JMD1340"
      ]
    }
  ],
  "5": [
    1,
    {
      "1": [
        "MYCUBE89212"
      ]
    }
  ],
  "6": [
    1,
    {
      "1": [
        "NEWHOLEFOODS"
      ]
    }
  ],
  "7": [
    1,
    {
      "1": [
        "PATIENTS"
      ]
    }
  ],
  "8": [
    1,
    {
      "1": [
        "PREBUILTHOLEFOODS"
      ]
    }
  ],
  "9": [
    8,
    {
      "1": [
        "RCITIES"
      ],
      "2": [
        "FILTERRDOCTORS"
      ],
      "3": [
        "RCITYRAINFALL"
      ],
      "4": [
        "RDOCTORS"
      ],
      "5": [
        "FILTERRPATIENTS"
      ],
      "6": [
        "RPATIENTS"
      ],
      "7": [
        "RPATIENTSDEPENDSON"
      ],
      "8": [
        "RPATIENTSINHERIT"
      ]
    }
  ],
  },
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/RepairBuild

Performs an unscheduled repair build of a registered cube.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

cubeName

Required. The name of the cube to repair.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/RepairBuild`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "cubeName": "Patients"
}
```

Example Response

```
{
  "buildStats": {
    "buildStats_1": {
      "caption": "Patients",
      "elapsedTime": 0.494022,
      "errors": 0,
      "event": "build",
      "eventId": "120",
      "expressionTime": 0.345886,
      "factCount": 1000,
      "iKnowTime": 0,
      "missingReferences": 0,
      "status": 1
    }
  },
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/SetActiveRegistry

Changes which cube registry is currently active.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint uses an empty request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/setActiveRegistry`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

`{}`

Example Response

```
{
```

While this API call returns an empty response, it also sets `^DeepSee.CubeManager("activeRegistry")` to `DeepSee.CubeManager.CubeRegistryDefinition`.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/SynchronizeCube

A wrapper for %DeepSee.Utils.%SynchronizeCube which records cube event information when building the cube.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

cubeName

Required. The name of the cube to synchronize.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Command/SynchronizeCube`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "cubeName": "Patients"
}
```

Example Response

```
{
  "factsUpdated": 0,
  "synchronizeStats": {
    "caption": "Patients",
    "event": "synchronize",
    "eventId": "132",
    "status": 1
  },
  "Error": "",
  "status": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Command/WriteToRegistry

Traverses a CubeManager.RegistryMap object and registers all the cubes.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following property of the request body:

cubeMap

Required. The name of the RegistryMap object to traverse.

Example Request

- *Request Method:*

POST

- *Request URL:*

https://localhost/api/deepsee/v2/Command/WriteToRegistry

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "cubeMap": "Groups"
}
```

Example Response

```
{
  "cubeMap":
  "{ \"Groups\": [ { \"Cubes\": [ { \"CubeKey\": \"EXPRESSIONHOLEFOODS\", \"CubeDisplayName\": \"HoleFoods
  2, \"Error\": \"\",
  \"status\": 1,
  \"tSC\": 1
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Data/GetDSTIME

Retrieves the last ^OBJ.DSTIME timestamp that the server processed for a given cube.

Request Method

GET

URL Parameters

sourceClass

Required. Full name of the source class of the cube.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

GET

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/GetDSTime/HoleFoods.Transation`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Status": "OK",
  "DispatchClass": "%DeepSee.REST.v2.InfoServer",
  "NameSpace": "SAMPLES"
}
```

/Data/KPIExecute

Executes the query defined by a KPI.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following properties of the request body:

KPI

Required. Logical name of the KPI.

FILTERS

Optional. Array of filter objects that specify how the KPI is filtered. Each filter object must provide the following properties:

- `name` — a filter specification such as `[aged].[h1].[age group]`
- `value` — logical name of a member of a filter, such as `&[0 to 29]`

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/KPIExecute`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "KPI": "DemoMDX",
  "FILTERS": [ { "name" : "[aged].[h1].[age group]", "value" : "&[0 to 29]" } ]
}
```

Example Response

```
{
  "Info": {
    "Error": "",
    "KpiName": "DemoMDX"
  },
  "Result": {
    "Properties": [
      { "name": "PatCount", "caption": "PatCount", "columnNo": 1 },
      { "name": "AvgTestScore", "caption": "AvgTestScore", "columnNo": 2 }
    ],
    "Series": [
      { "PatCount": 482, "AvgTestScore": 73.62564102564102564, "seriesName": "Cedar Falls" },
      { "PatCount": 473, "AvgTestScore": 74.54089709762532982, "seriesName": "Centerville" },
      { "PatCount": 454, "AvgTestScore": 73.86532951289398281, "seriesName": "Cypress" },
    ]
  }
}
```

```

    {
      "PatCount": 471, "AvgTestScore": 75.69459459459459459, "seriesName": "Elm Heights"},
      {
        "PatCount": 468, "AvgTestScore": 74.00806451612903226, "seriesName": "Juniper"},
        {
          "PatCount": 464, "AvgTestScore": 73.71925133689839572, "seriesName": "Magnolia"},
          {
            "PatCount": 438, "AvgTestScore": 73.76123595505617978, "seriesName": "Pine"},
            {
              "PatCount": 464, "AvgTestScore": 75.46537396121883657, "seriesName": "Redwood"},
              {
                "PatCount": 445, "AvgTestScore": 75.19886363636363636, "seriesName": "Spruce"}
            }
          }
        }
      }
    }
  ]
}

```

In the response object, the `Result` property contains the properties `Series` and `Properties`. The `Series` property contains an array of objects, one for each series (or row) in the KPI. The `Properties` property contains an array of objects, one for each row in the KPI.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Data/MDXCancelQuery/:queryID

Cancels a previously started query, given the ID of the query.

Request Method

POST

URL Parameters

queryID

Required. ID of the query to cancel. If you started the query with the [POST /Data/MDXExecute](#) endpoint, obtain the ID of the query from the `Info.QueryID` property of the response object returned by that endpoint. If you started the query with the [POST /Data/PivotExecute](#) endpoint, obtain the ID of the query from the `Info.QueryID` property of the response object returned by that endpoint.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/MDXCancelQuery/:patients|en2515296118`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

/Data/MDXDrillthrough

Executes a detail listing.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint uses the following properties of the request body:

MDX

Required. MDX SELECT QUERY, preceded by either DRILLTHROUGH or DRILLFACTS.

Use DRILLTHROUGH to use a named detail listing or to retrieve fields from the source class of the cube. Use DRILLFACTS to retrieve fields from the fact table.

If the base SELECT query returns more than one cell, only the top left cell is used for the detail listing.

LISTING

Optional. Logical name of the detail listing to use. You must specify either LISTING or RETURN, but not both.

WAIT

Optional. Specify 0 or 1 (the default). If this property is 0, the server sends partial results. If this property is 1, the server assumes the client wishes to wait for complete results before sending a response.

RETURN

Optional. List of fields in the applicable table, depending on the value in MDX. If you specify this, the listing consists of these fields.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/MDXDrillthrough`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "MDX" : "DRILLTHROUGH SELECT FROM patients WHERE AGED.60",
  "LISTING" : "Patient details"
}
```

For another example:

```
{
  "MDX" : "DRILLTHROUGH SELECT FROM patients WHERE AGED.60",
  "RETURN": "Age,BirthDate"
}
```

For another example:

```
{
  "MDX" : "DRILLFACTS SELECT FROM patients WHERE AGED.60",
  "RETURN": "MxAge,MxTestScore"
}
```

Example Response

```
{
  "Info": {
    "Error": "",
    "MDXText": "DRILLTHROUGH SELECT FROM [PATIENTS] WHERE [AGED].[60]",
    "QueryKey": "en2156087935",
    "CubeKey": "PATIENTS",
    "QueryID": "PATIENTS|en2156087935",
    "Cube": "patients",
    "ResultsComplete": 1,
    "Pivot": "",
    "QueryType": "DRILLTHROUGH",
    "ListingSource": "source",
    "ColCount": 5,
    "RowCount": 0,
    "Error": "",
    "TimeStamp": "2016-08-14 15:43:04"
  },
  "AxesInfo": [
    { "%ID": "SlicerInfo", "Text": "[AGED].[60]" },
    { "%ID": "AxisInfo_1", "Text": "[%SEARCH]" },
    { "%ID": "AxisInfo_2", "Text": "[%SEARCH]" }
  ],
  "Result": { "children": [
    { "PatientID": "SUBJ_100508", "Age": 60, "Gender": "Female", "Home City": "Elm Heights", "Test Score": 81 },
    { "PatientID": "SUBJ_100539", "Age": 60, "Gender": "Female", "Home City": "Elm Heights", "Test Score": 90 },
    { "PatientID": "SUBJ_100701", "Age": 60, "Gender": "Female", "Home City": "Redwood", "Test Score": 61 },
    { "PatientID": "SUBJ_100829", "Age": 60, "Gender": "Female", "Home City": "Juniper", "Test Score": 98 },
    ... ]
  }
}
```

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Data/MDXExecute

Executes an MDX query and obtains the results.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following properties of the request body:

MDX

Required. MDX SELECT QUERY.

FILTERS

Optional. Any additional filters to add to the query. If specified, this property must be an array of strings, each of which specifies a filter value.

WAIT

Optional. Specify 0 or 1 (the default). If this property is 0, the server sends partial results. If this property is 1, the server assumes the client wishes to wait for complete results before sending a response.

TIMEOUT

Optional. Timeout for waiting for query results, in seconds. The default timeout for this wait is 2 seconds less than the session's timeout setting.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/MDXExecute`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{"MDX": "SELECT aged.[age group].MEMBERS ON 0 FROM PATIENTS"}
```

For another example:

```
{"MDX": "SELECT FROM PATIENTS"}
```

For another example:

```
{"MDX": "SELECT birthd.date.members on 0 FROM PATIENTS", "WAIT":1, "TIMEOUT":30}
```

For another example:

```
{
  "MDX": "SELECT FROM PATIENTS",
  "FILTERS": [ "[HomeD].[H1].[ZIP].&[32006]", "[HomeD].[H1].[ZIP].&[32007]" ],
  "WAIT": 1,
  "TIMEOUT": 30
}
```

Example Response

Response for MDX Query

Note that [POST /Data/PivotExecute](#) and [POST /Data/MDXUpdateResults](#) return the same response body.

The `Info.QueryID` property contains the query ID, which you need as input for the [POST /Data/MDXCancelQuery](#) and [POST /Data/MDXUpdateResults](#) services. An `Info.ResultsComplete` property with a value of 1 indicates that the MDX query has completed. Note that if the `Info.PendingResults` property has a value of 1, plugins are still computing, although the rest of the query may have completed. An `Info.PendingResults` property with a value of 0 indicates that any plugins have finished computing.

For information that applies to all response objects, see [Notes on the Response Objects](#).

```
{
  "Info": {
    "Error": "",
    "MDXText": "SELECT [AGED].[AGE GROUP].MEMBERS ON 0 FROM [PATIENTS]",
    "QueryKey": "en2772997983",
    "CubeKey": "PATIENTS",
    "QueryID": "PATIENTS|en2772997983",
    "Cube": "PATIENTS",
    "ResultsComplete": 1,
    "Pivot": "",
    "QueryType": "SELECT",
    "ListingSource": "",
    "ColCount": 3,
    "RowCount": 0,
    "TimeStamp": "2016-08-14 16:05:16"
  },
  "AxesInfo": [
    { "%ID": "SlicerInfo", "Text": "" },
    { "%ID": "AxisInfo_1", "Text": "[AGED].[AGE GROUP].MEMBERS" },
    { "%ID": "AxisInfo_2", "Text": "[%SEARCH]" }
  ],
  "Result": {
    "Axes": [
      { "%ID": "Axis_1", "Tuples": [
        { "%ID": "Tuple_1",
          "Members": [
            { "%ID": "Member_1", "Name": "0 to 29" }
          ]
        },
        { "%ID": "MemberInfo": [
          { "%ID": "MemberInfo_1",
            "nodeNo": 3, "text": "0 to 29",
            "dimName": "AgeD",
            "hierName": "H1",
            "levelName": "Age Group",
            "memberKey": "0 to 29",
            "dimNo": 2,
            "hierNo": 1,
            "levelNo": 2,
            "aggregate": "",
            "orSpec": ""
          }
        ]
      }
    ],
    "TupleInfo": [
      { "%ID": "TupleInfo_1", "childSpec": "[AgeD].[H1].[Age Group].&[0 to 29].children" },
      { "%ID": "TupleInfo_2", "childSpec": "[AgeD].[H1].[Age Group].&[30 to 59].children" },
      ...
    ],
    "CellData": [
      { "%ID": "Cell_1", "ValueLogical": 418, "Format": "", "ValueFormatted": "418" },
      { "%ID": "Cell_2", "ValueLogical": 421, "Format": "", "ValueFormatted": "421" },
      ...
    ]
  }
}
```

```

    ...]
  }
}

```

If the response is incomplete, it includes cell data objects like the following:

```

{"%ID": "Cell_9", "ValueLogical": "@Computing
  0.00%", "Format": "", "ValueFormatted": "@Computing 0.00%"}

```

Response for MDX DRILLTHROUGH Query

```

{
  "Info": {
    "Error": "",
    "TimeStamp": "2017-09-26 15:31:23",
    "ResultsComplete": 1,
    "MDXText": "DRILLTHROUGH SELECT [AGED].[AGE GROUP].[0 TO 29] ON 0 FROM [PATIENTS]",
    "QueryKey": "en2983351588",
    "CubeKey": "PATIENTS",
    "QueryID": "PATIENTS|en2983351588",
    "Cube": "PATIENTS",
    "Pivot": "",
    "QueryType": "DRILLTHROUGH",
    "ListingSource": "source",
    "ColCount": 5,
    "RowCount": 0
  },
  "AxesInfo": [
    { "%ID": "SlicerInfo",
      "Text": "" },
    { "%ID": "AxisInfo_1",
      "Text": "[AGED].[AGE GROUP].[0 TO 29]" },
    { "%ID": "AxisInfo_2",
      "Text": "[%SEARCH]" }
  ],
  "Result": {
    "children": [
      { "PatientID": "SUBJ_100786",
        "Age": 0,
        "Gender": "Female",
        "Home City": "Centerville",
        "Test Score": 77 },
      { "PatientID": "SUBJ_100960",
        "Age": 0,
        "Gender": "Female",
        "Home City": "Elm Heights",
        "Test Score": 62 },
      { "PatientID": "SUBJ_100977",
        "Age": 0,
        "Gender": "Female",
        "Home City": "Elm Heights",
        "Test Score": 54 },
      ... ]
    }
  }
}

```

/Data/MDXUpdateResults/:queryID

Retrieves updated results for a given query that was previously incomplete.

Request Method

POST

URL Parameters

queryID

Required. ID of the query. If you started the query with the [POST /Data/MDXExecute](#) endpoint, obtain the ID of the query from the `Info.QueryID` property of the response object returned by that endpoint. If you started the query with the [POST /Data/PivotExecute](#) endpoint, obtain the ID of the query from the `Info.QueryID` property of the response object returned by that endpoint.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/MDXCancelQuery/:patients|en2515296118`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

This endpoint returns the same response object as [POST /Data/MDXExecute](#).

/Data/PivotExecute

Executes the MDX query defined by a pivot table and obtains the results.

Request Method

POST

URL Parameters

None. Note that a request body is required; see the next heading.

Request Body Details

This endpoint uses the following properties of the request body:

PIVOT

Required. Full logical name of the pivot table, including the name of the folder that contains it.

FILTERS

Optional. Specifies any additional filters to add to the query. If specified, this property must be an array of strings, each of which specifies a filter value.

WAIT

Optional. Specify 0 or 1 (the default). If this property is 0, the server sends partial results. If this property is 1, the server assumes the client wishes to wait for complete results before sending a response.

TIMEOUT

Optional. Timeout for waiting for query results, in seconds. The default timeout for this wait is 2 seconds less than the session's timeout setting.

VARIABLES

Optional. Specifies the values for any pivot variables used in the pivot table. Specify this as an array of objects. Each object must specify the name and value properties.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/PivotExecute`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{ "PIVOT": "Use in Dashboards/Product Info" }
```

For another example:

```
{ "PIVOT": "Pivot Variables/Commission Calculator",
  "VARIABLES": [ { "name": "commissionpercentage", "value": 15 } ]
}
```

Example Response

This endpoint returns the same response object as [POST /Data/MDXExecute](#).

/Data/TestConnection

Tests the connection to the server.

Request Method

GET

URL Parameters

None.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

GET

- *Request URL:*

`https://localhost/api/deepsee/v2/Data/TestConnection`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Status": "OK",
  "DispatchClass": "%DeepSee.REST.v2.DataServer",
  "NameSpace": "SAMPLES"
}
```

/Info/Cubes

Returns information about the cubes and subject areas available in the InterSystems IRIS® data platform namespace that you access via this REST call.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint uses the following properties of the request body:

TYPE

Optional. This property can be `cubes` or `subjectareas`. If this property is `cubes`, the server sends information only about cubes. If this property is `subjectareas`, the server sends information only about subject areas. If this property is not specified, the server sends information about both cubes and subject areas.

BASECUBE

Optional. If specified, this property should equal the logical name of a cube. In this case, the server sends information only about cubes and subject areas based on this cube.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/Cubes`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{"BASECUBE": "patients"}
```

Example Response

```
{
  "Info": {
    "Error": "",
    "Type": "all",
    "BaseCube": "patients"
  },
  "Result": {
    "Cubes": [
      {
        "name": "AsthmaPatients", "displayName": "AsthmaPatients", "modTime": "2016-11-14
20:49:14", "type": "subjectArea"},
      {
        "name": "DemoMDX", "displayName": "DemoMDX", "modTime": "2016-11-14
20:49:14", "type": "subjectArea"},
      {
        "name": "YoungPatients", "displayName": "YoungPatients", "modTime": "2016-11-14
20:49:14", "type": "subjectArea"}
    ]
  }
}
```


In the response object, the `Result` property contains a property called `Cubes`, which contains an array of objects, one for each cube or subject area. In these objects, the `type` property indicates whether the item is a cube or a subject area.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/Dashboards

Returns information about the dashboards available in the InterSystems IRIS namespace that you access via this REST call.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/Dashboards`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": ""
  },
  "Result": {
    "Dashboards": [
      {
        "fullName": "Basic Dashboard Demo",
        "name": "Basic Dashboard Demo",
        "lastModified": "2016-11-14 19:39:14",
        "itemType": "dashboard"
      },
      {
        "fullName": "Custom Drilldown Spec",
        "name": "Custom Drilldown Spec",
        "lastModified": "2016-11-14 19:39:14",
        "itemType": "dashboard"
      }
    ]
  }
}
```

In the response object, the `Result` property contains a property called `Dashboards`, which contains an array of objects, one for each dashboard.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/FilterMembers/:datasource/:filterSpec

Returns information about the members of the given filter, as defined by the given data source (which is either a cube or a KPI).

Request Method

POST

URL Parameters

datasource

Required. Name of the data source. This is one of the following:

- *cubename* — a logical cube name
- *cubename.cube* — a logical cube name, followed by *.cube*
- *kpiname.kpi* — a logical KPI name, followed by *.kpi*

This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

filterSpec

Required. Filter specification.

Request Body Details

This endpoint uses the following properties of the request body. These properties both specify filters that are applied to the data source, thus limiting the list of members returned by the endpoint.

RELATED

Optional. If specified, this property is an array of objects, each of which contains the *spec* property (a filter specification) and the *value* property (value of that filter). A *value* property should be an MDX set expression and should use member keys.

SEARCHKEY

Optional.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/FilterMembers/:demomdx.kpi/:homed.h1.zip`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- *Request Body:*

```
{
  "RELATED": [
    { "spec": "gend.h1.gender", "value": "&[female]" }
  ],
  "SEARCHKEY": "Jan"
}
```

Example Response

```
{
  "Info": {
    "Error": "",
    "DataSource": "demomdx.cube",
    "DataSourceType": "cube",
    "Default": "",
    "Filter": "[HomeD].[H1].[ZIP]"
  },
  "Result": {
    "FilterMembers": [
      {
        "text": "32006", "value": "&[32006]", "description": ""
      },
      {
        "text": "32007", "value": "&[32007]", "description": ""
      },
      {
        "text": "34577", "value": "&[34577]", "description": ""
      },
      {
        "text": "36711", "value": "&[36711]", "description": ""
      },
      {
        "text": "38928", "value": "&[38928]", "description": ""
      }
    ]
  }
}
```

In the response object, the `Result` property contains a property called `FilterMembers`, which contains an array of objects, one for each filter member. The object for a given filter member contains the following properties:

- `description` contains the text description of the member, if any
- `text` contains the display text for the member
- `value` contains the logical value of the member (typically the MDX key)

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/Filters/:datasource

Returns information about the filters available for the given data source (which is either a cube or a KPI).

Request Method

POST

URL Parameters

datasource

Required. Name of the data source. This is one of the following:

- *cube*name — a logical cube name
- *cube*name.cube — a logical cube name, followed by .cube
- *kpi*name.kpi — a logical KPI name, followed by .kpi

This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/Filters/:aviationevents`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": "",
    "DataSource": "aviationevents.cube",
    "DataSourceType": "cube"
  },
  "Result": {
    "Filters": [
      {
        "caption": "Year", "value": "[EventDateD].[H1].[Year]", "type": "year",
      },
      {
        "caption": "Month", "value": "[EventDateD].[H1].[Month]", "type": "" ,
      },
      {
        "caption": "Day", "value": "[EventDateD].[H1].[Day]", "type": ""
      },
      ...
    ]
  }
}
```

In the response object, the `Result` property contains a property called `Filters`, which contains an array of objects, one for each filter. Each object has the following properties:

- `caption` contains the display value for the filter.
- `type` contains the filter type, if it exists. This can be `"calc"` or can be the name of a time function. In other cases it is empty.

- `value` contains the filter specification, which is the logical identifier for the filter. For information on the filter specification, see [POST /Info/FilterMembers/:datasource/:filterSpec](#).

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/KPIs

Returns information about the KPIs available in the InterSystems IRIS namespace that you access via this REST call.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/KPIs`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": ""
  },
  "Result": {
    "KPIs": [
      {
        "name": "Aviation Actions", "caption": "Aviation Actions", "moddate": "2016-11-14
11:22:08", "type": "kpi"},
      {
        "name": "AviationTopConcepts", "caption": "AviationTopConcepts", "moddate": "2016-11-14
11:22:08", "type": "kpi"},
      {
        "name": "BPDdiastolic", "caption": "BPDdiastolic", "moddate": "2016-11-14 11:22:08", "type": "kpi"}
      ...]
    }
  }
}
```

In the response object, the `Result` property contains a property called `KPIs`, which contains an array of objects, one for each KPI.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/ListingFields/:cube

Returns information about the <listingField> elements available in the InterSystems IRIS namespace that you access via this REST call.

Request Method

POST

URL Parameters

cube

Required. Logical name of the cube. This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/ListingFields/:holefoods`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": "",
    "BaseCube": "holefoods"
  },
  "Result": {
    "ListingFields": [
      { "caption": "Channel", "expression": "%EXTERNAL(Channel) Channel" },
      { "caption": "City", "expression": "Outlet->City" },
      { "caption": "Comment", "expression": "Comment" },
      ... ]
    }
  }
```

In the response object, the `Result` property contains a property called `ListingFields`, which contains an array of objects, one for each <listingField> element.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/Listings/:cube

Returns information about the detail listings available for the given cube.

Request Method

POST

URL Parameters

cube

Required. Logical name of the cube. This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

Request Body Details

This endpoint uses the following property of the request body:

TYPE

Optional. This property can be `map` or `table`. If this property is `map`, the server sends information only about map listings. If this property is `table`, the server sends information only about non-map listings. If this property is not specified, the server sends information about all kinds of listings.

Example Request

- Request Method:*

POST

- Request URL:*

`https://localhost/api/deepsee/v2/Info/Listings/:holefoods`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- Request Body:*

```
{ "TYPE": "map" }
```

Example Response

For another example:

```
{
  "Info": {
    "Error": "",
    "DataSource": "holefoods",
    "Type": "all"
  },
  "Result": {
    "Listings": [
      {
        "name": "Custom Listing",
        {
          "name": "Another Sample Listing by Date",
          "fields": "%ID As \"ID #\", DateOfSale As \"Sale Date\"",
          "order": "DateOfSale,%ID",
          "type": "table",
          "source": "listingGroup",
          "edit": 1,
        },
      {
        "name": "Another Sample Listing with Customer Info",
        "fields": "%ID,Outlet->City \"Store Location\",Outlet->Country->Name Country,Product->Name Product,ZipCode \"Customer ZipCode\",Latitude,Longitude",
        "order": "",
        "type": "map",
        "source": "listingGroup",
        "edit": 1,
      }
    ]
  }
}
```

```
        {"name": "Customer Info",
         "fields": "%ID,Outlet->City \"Store Location\",Outlet->Country->Name Country,Product->Name
Product,ZipCode \"Customer ZipCode\",Latitude,Longitude", "order": "",
         "type": "map",
         "source": "cube",
         "edit": 0},
        ...]
    }
}
```

In the response object, the `Result` property contains a property called `Listings`, which contains an array of objects, one for each detail listing.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/Measures/:cube

Returns information about the measures available to the given cube.

Request Method

POST

URL Parameters

cube

Required. Logical name of the cube. This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

Request Body Details

This endpoint uses the following property of the request body:

SKIPCALCULATED

Optional.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/Measures/:demomdx`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": "",
    "BaseCube": "DemoMDX",
    "SkipCalculated": 0
  },
  "Result": {
    "Measures": [
      { "name": "%COUNT", "caption": "%COUNT", "type": "integer", "hidden": 0, "factName": "" },
      { "name": "Age", "caption": "Age", "type": "integer", "hidden": 0, "factName": "MxAge" }
      ...
    ]
  }
}
```

In the response object, the `Result` property contains a property called `Measures`, which contains an array of objects, one for each measure.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/NamedFilters/:cube

Returns information about the named filters available to the given cube.

Request Method

POST

URL Parameters

cube

Required. Logical name of the cube. This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/NamedFilters/:holefoods`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": "",
    "BaseCube": "holefoods"
  },
  "Result": {
    "NamedFilters": [
      {
        "name": "Sample Named Filter",
        "description": "",
        "spec": "[Product].[P1].[Product Category].&[Seafood]",
        "cube": "HOLEFOODS"
      }
    ]
  }
}
```

In the response object, the `Result` property contains a property called `NamedFilters`, which contains an array of objects, one for each named filter.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/Pivots

Returns information about the pivot tables available in the InterSystems IRIS namespace that you access via this REST call.

Request Method

POST

URL Parameters

None.

Request Body Details

This endpoint uses the following property of the request body:

BASECUBE

Optional. If specified, this property should equal the logical name of a cube. In this case, the server sends information only about pivot tables based on this cube.

Example Request

- Request Method:*

POST

- Request URL:*

`https://localhost/api/deepsee/v2/Info/Pivots`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

- Request Body:*

```
{ "BASECUBE": "PATIENTS" }
```

Example Response

```
{
  "Info": {
    "Error": "",
    "BaseCube": ""
  },
  "Result": {
    "Pivots": [
      {
        "fullName": "Calculated Members\\Alternative Avg Allergy Count",
        "name": "Alternative Avg Allergy Count",
        "lastModified": "2016-11-14 11:22:08",
        "itemType": "pivot"
      },
      {
        "fullName": "Calculated Members\\Average Patient Count per Decade",
        "name": "Average Patient Count per Decade",
        "lastModified": "2016-11-14 11:22:08",
        "itemType": "pivot"
      }
    ]
  }
}
```

In the response object, the `Result` property contains a property called `Pivots`, which contains an array of objects, one for each pivot table.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/PivotVariableDetails/:cube/:variable

Returns detailed information about the given pivot variable.

Request Method

POST

URL Parameters

cube

Required. Logical name of a cube that has access to the given pivot variable. This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

variable

Required. Logical name of the pivot variable.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/PivotVariableDetails/:holefoods/:Year`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": "",
    "BaseCube": "holefoods",
    "Variable": "Year"
  },
  "Result": {
    "Details": {
      "context": "literal",
      "defaultValue": "NOW",
      "description": "",
      "displayList": "",
      "displayName": "Year",
      "name": "Year",
      "sourceName": "HoleFoodsYears.kpi",
      "sourceType": "kpi",
      "type": "string",
      "valueList": ""
    }
  }
}
```

In the response object, the `Result` property contains a property called `Details`, which contains the details for the pivot variable.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/PivotVariables/:cube

Returns information about the pivot variables available to the given cube.

Request Method

POST

URL Parameters

cube

Required. Logical name of the cube. This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/PivotVariables/:holefoods`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": "",
    "BaseCube": "holefoods"
  },
  "Result": {
    "PivotVariables": [
      {
        "name": "CommissionPercentage",
        "caption": "Commission Percentage",
        "defValue": "0",
        "context": "literal",
        "desc": ""
      },
      {
        "name": "Year",
        "caption": "Year",
        "defValue": "NOW",
        "context": "literal",
        "desc": ""
      }
    ]
  }
}
```

In the response object, the `Result` property contains a property called `PivotVariables`, which contains an array of objects, one for each pivot variable.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/QualityMeasures/:cube

Returns information about the quality measures available to the given cube.

Request Method

POST

URL Parameters

cube

Required. Logical name of the cube. This name can include slashes; see [Use of Slashes in Cube and KPI Names](#).

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

POST

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/QualityMeasures/:holefoods`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Info": {
    "Error": "",
    "BaseCube": "holefoods"
  },
  "Result": {
    "QualityMeasures": [
      {
        "name": "TestCatalog\\TestSet\\TestQM",
        "caption": "Sample Quality Measure",
        "description": ""
      }
    ]
  }
}
```

In the response object, the `Result` property contains a property called `QualityMeasures`, which contains an array of objects, one for each quality measure.

For information that applies to all response objects, see [Notes on the Response Objects](#).

/Info/TestConnection

Tests the connection to the server.

Request Method

GET

URL Parameters

None.

Request Body Details

This endpoint ignores the request body.

Example Request

- *Request Method:*

GET

- *Request URL:*

`https://localhost/api/deepsee/v2/Info/TestConnection`

For comments on the possible forms of the URL, see [Introduction to the Business Intelligence REST API](#).

Example Response

```
{
  "Status": "OK",
  "DispatchClass": "%DeepSee.REST.v2.InfoServer",
  "NameSpace": "SAMPLES"
}
```

