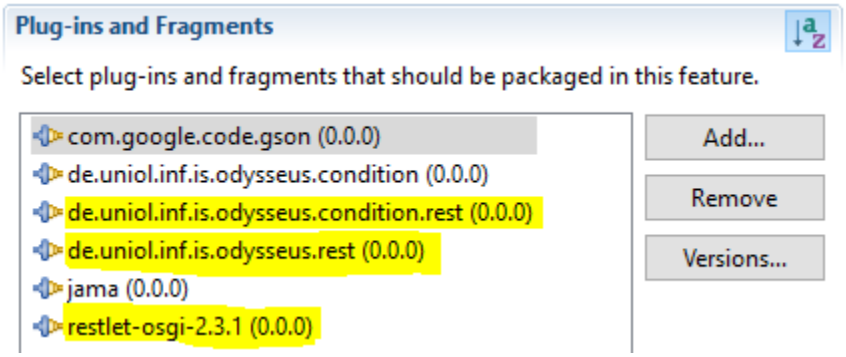
REST interface

Odysseus provides a REST interface to install and control queries and to get the results of queries. Please install the REST feature to use it.

Using the REST interface when developing with Odysseus

The REST interface uses RESTlet. A feature which wants to provide plug-ins which offer REST interfaces needs to add the osgi restlet version to the plug-ins.The list of plug-ins of the condition monitoring feature is shown in the figure below.



Using the REST interface

Client library

If you want to create your own client (in Java) to communicate with Odysseus but you want to reduce the workload for the communication, you can use the Odysseus Client to do the communication for you (at least, partly). You can find the project here: https://git.offis.uni-oldenburg.de/projects/ODY/repos/odysseus-client/commits

Of course, it is also possible to use the REST service in a standard way. Please use application/json as content of a POST request to talk with Odysseus. For an example see "login" below.

Address

To use the REST interface from a remote application, the right address is needed. The address is build by the following schema: BASE_PROTOCOL + ip + ":" + SERVICE_PORT + "/" + SERVICE_PATH + "/" + RESOURCE_PATH

The variables need to be filled with the right strings from the table below. The variables SERVICE_PATH and RESOURCE_PATH need to be filled for the specific interface you want to use. The values are listed for each service / resource in the sections below.

Variable	Value
BASE_PROTOCOL	`http://`
SERVICE_PORT	`9679`
ip	The IP address of you Odysseus instance (e.g. 127.0.0.1 if located on the same machine as the client)

Remark:The Odysseus server tries to find a free port. If 9679 is blocked it will use 9680 etc. In case your connection fails, please look in server log to see something like this:

```
Feb 21, 2017 4:15:14 PM org.restlet.engine.connector.NetServerHelper start
INFO: Starting the internal [HTTP/1.1] server on port 9679
157
 INFO  RestService  - Restservice published at 9679 -
de.uniol.inf.is.odysseus.rest.service.RestService.start(RestService.java:52)
```

Communication objects

To communicate with the REST interface, you have to send objects to and receive objects from Odysseus. If you are developing in Java, the easiest way to make sure you use correct objects for communication is to copy the data transfer objects ("DTO") from the Odysseus source code to your project. You can find these objects in the REST plug-in in the package "dto". E.g. "AbstractLoginRequestDTO" and so on.

The core REST interface

Variable	Value
SERVICE_PATH	`core`

The core REST interface provides functionality to login, install queries, start and stop them and to create sockets to get the results of the queries. The answer of the server is always a "GenericResponseDTO" (DTO stands for Data Transfer Object) which is basically an envelope for the data. It's useful to have such an object on the client-side to cast the object. You can use the code for the GenericResponseDTO from the codeblock below:

GenericResponseDTO

```
public class GenericResponseDTO<T> {

    private T value;

    public GenericResponseDTO(T value) {
        this.value = value;
    }

    public GenericResponseDTO() {}

    public T getValue() {
        return value;
    }

    public void setValue(T value) {
        this.value = value;
    }
}
```

Login

Variable	Value
RESOURCE_PATH	`login`

You have to send an object with the following fields:

Name	Type
username	String
password	String
tenant	String

For everything you want to do with the Odysseus REST interface you need a security token which you can get by logging in. The token is valid as long as the session is active - normally until Odysseus is restarted. To login you need so send a LoginRequest-Object, which is basically an object with username, password and tenant as strings. For example, you could fill this with the standard information "System" and "manager" and an empty tenant. The answer of the server contains the token for this user, if the login information was correct. The code below shows an example implementation in Java to login to Odysseus:

Login method

```
/**
 * Logs in to Odysseus with a specific username and password
 *
 * @param ip       IP of the Odysseus instance
 * @param username Username of the Odysseus instance (e.g. "System")
 * @param password Password of the given user (e.g. "manager")
 * @return A token which you can use to send requests
 * @throws RestException
 */
public static String login(String ip, String username, String password) throws RestException {
    String hostURL = BASE_PROTOCOL + ip + ":" + SERVICE_PORT + "/" + SERVICE_PATH_CORE + "/" +
RESOURCE_PATH_LOGIN;
    ClientResource cr = new ClientResource(hostURL);
        String tenant = "";
        LoginRequestDTO req = new LoginRequestDTO(username, password, tenant);

    try {
        Representation t = cr.post(req);
        Type resultDataType = new TypeToken<GenericResponseDTO<String>>() {
        }.getType();
        Gson gson = new Gson();
        GenericResponseDTO<String> resp = gson.fromJson(t.getText(), resultDataType);
        return resp.getValue();
    } catch (Exception ex) {
        throw new RestException(ex.toString());
    } finally {
        cr.release();
    }
}
```

Pure REST

POST-REQUEST: http://localhost:9679/core/login

with content:

```
{
    "username" : "System",
    "password" : "manager",
    "tenant" : ""
}
```

Response (if success), something like:

```
{
    "value" : "j983q1c0ktqtab65jja8bt9s2"
}
```

This token must be used in other requests with attribute token.

```
{
    "token" : "j983q1c0ktqtab65jja8bt9s2",
    "other" : "value"
 }
```

The other pure REST calls are similar.

Add a query

With this resource you can add queries to Odysseus.

Variable	Value
RESOURCE_PATH	`addQuery`

You have to send an object with the following fields:

Name	Type	Explanation
token	String	From the login
query	String	E.g. a PQL query
parser	String	E.g. "OdysseusScript" or "PQL"
transformationConfig	String	Can be empty

The answer contains a list (collection) of integers. These integers are the queryIds of the added queries.

If you use "OdysseusScript" as parser and use "#RUNQUERY", you can directly start the query with this resource.

Method to add a query via the Odysseus REST interface

```
/**
 *
 * @param ip The IP of the odysseus instance
 * @param token The security token from the login
 * @param query The query test
 * @return A map with the name of the operator -> the output port of this operator -> the socketInformation
 * @throws RestException
 */
public static Map<String, Map<Integer, SocketInfo>> runQuery(String ip, String token, String query) throws
RestException {
    String hostURL = BASE_PROTOCOL + ip + ":" + SERVICE_PORT + "/" + SERVICE_PATH_CORE;
    ClientResource crAddQuery = new ClientResource(hostURL + "/" + RESOURCE_PATH_ADD_QUERY);
    ClientResource crCreateSocket = new ClientResource(hostURL + "/" + RESOURCE_PATH_CREATE_SOCKET);

    Gson gson = new Gson();

    // Add query
    String transformationConfig = "";
    AddQueryRequestDTO addQueryRequestDTO = new AddQueryRequestDTO(token, query, "OdysseusScript",
transformationConfig);
    Representation addQueryRepresentation = crAddQuery.post(addQueryRequestDTO);
    GenericResponseDTO<Collection<Double>> queryIds = null;
    try {
        queryIds = gson.fromJson(addQueryRepresentation.getText(), GenericResponseDTO.class);
        // Create socket
        int queryId = queryIds.getValue().iterator().next().intValue();
        return getResultsFromQuery(ip, token, queryId);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

Start a query

With this resource you can start an existing query.

Variable	Value
RESOURCE_PATH	`startQuery`

You have to send an object with the following fields:

Name	Type	Explanation
token	String	From the login
value	Integer	The id of the query

The answer contains a boolean which is true, if the query is running.

Stop a query

With this resource you can stop an existing query.

Variable	Value
RESOURCE_PATH	`stopQuery`

You have to send an object with the following fields:

Name	Type	Explanation
token	String	From the login
value	Integer	The id of the query

The answer contains a boolean which is true, if the query is inactive.

Remove a query

With this resource you can remove an existing query.

Variable	Value
RESOURCE_PATH	`removeQuery`

You have to send an object with the following fields:

Name	Type	Explanation
token	String	From the login
value	Integer	The id of the query

The answer contains a boolean which is true, if the query is undefined (successfully removed).

Create a socket

With this resource you can create a sockets to get the results of the queries.

Variable	Value
RESOURCE_PATH	`createMultiSocket`

You have to send an object with the following fields:

Name	Type	Explanation
token	String	From the login
rootPort	Integer	Not used, can be empty for this resource
queryId	Integer	The id of the query you want to get the results from
queryName	String	The name of the query you want to get the results from
useQueryName	boolean	If true, the 'queryName' is used, if false, the 'queryId' is used

operatorName	String	The name of the operator you want to get the results from. Use, if you know the name of the output operator. If you don't set this, you will get the sockets of all output operators of the query.
outputOperatorPort	int	The output port of the output operator (one operator can have more than one output port)
useOutputOperatorPort	boolean	If true, the 'outputOperatorPort' is used, if false, all output operator ports are used.

The answer is not an GenericResponseDTO. The answer-object is a map with the following structure: Map<String, Map<Integer, SocketInfo>>. The outer map has the operator-names as key. The inner map has the output port as key. The code below gives an example of how to use this resource.

Method to get socket information from Odysseus

```
/**
 *
 * @param ip The IP of the Odysseus instance
 * @param createSocketRequestDTO The request object
 * @return  The outer map has the name of the operator as key, the inner map
 *          the output port of the operator
 * @throws RestException
 */
public static Map<String, Map<Integer, SocketInfo>> getResultsFromQuery(String ip, CreateSocketRequestDTO
createSocketRequestDTO) throws RestException {
    String hostURL = BASE_PROTOCOL + ip + ":" + SERVICE_PORT + "/" + SERVICE_PATH_CORE;
    ClientResource crCreateSocket = new ClientResource(hostURL + "/" + RESOURCE_PATH_CREATE_SOCKET);

    Gson gson = new GsonBuilder().registerTypeAdapter(SocketInfo.class, new SocketInfoDeserializer()).create();
    try {
        Representation createSocketRepresentation = crCreateSocket.post(createSocketRequestDTO);
        Type socketType = new TypeToken<HashMap<String, HashMap<Integer, SocketInfo>>>(){}.getType();
        return gson.fromJson(createSocketRepresentation.getText(), socketType);
    } catch (IOException ex) {
        throw new RestException(ex.toString());
    } finally {
        crCreateSocket.release();
    }
}
```

RunCommand

Variable	Value
RESOURCE_PATH	`runCommand`

Name	Type	Explanation
token	String	From the login
value	String	command

Where command is an expression with a Command.

GetAllOperatorInformation

Variable	Value
RESOURCE_PATH	`getAllOperatorInformation`

Retrieve all currently available operators.

Name	Type	Explanation
token	String	From the login

GetOperatorInformation

Variable	Value
RESOURCE_PATH	`getOperatorInformation`

Retrieve information for a single operator

Name	Type	Explanation
token	String	From the login
value	String	operator name

GetViews

Variable	Value
RESOURCE_PATH	`getViews`

Get information about all currently installed views and sources

Name	Type	Explanation
token	String	From the login


```
[
  {
    "name": {
      "resourceName": "nexmark:auction",
      "user": "System"
    },
    "schema": {
      "attributes": [
        {
          "sourcename": "auction",
          "attributename": "timestamp",
          "datatype": {
            "uri": "StartTimestamp",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
        {
          "sourcename": "auction",
          "attributename": "id",
          "datatype": {
            "uri": "Integer",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
        {
          "sourcename": "auction",
          "attributename": "itemname",
          "datatype": {
            "uri": "String",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
```

```
    },
    {
      "sourcename": "auction",
      "attributename": "description",
      "datatype": {
        "uri": "String",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "auction",
      "attributename": "initialbid",
      "datatype": {
        "uri": "Integer",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "auction",
      "attributename": "reserve",
      "datatype": {
        "uri": "Integer",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "auction",
      "attributename": "expires",
      "datatype": {
        "uri": "Long",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "auction",
      "attributename": "seller",
      "datatype": {
        "uri": "Integer",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "auction",
      "attributename": "category",
      "datatype": {
        "uri": "Integer",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    }
  ],
  "uri": "auction",
  "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
}
```

```
},
{
  "name": {
    "resourceName": "nexmark:bid",
    "user": "System"
  },
  "schema": {
    "attributes": [
      {
        "sourcename": "bid",
        "attributename": "timestamp",
        "datatype": {
          "uri": "StartTimestamp",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "auction",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "bidder",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "datetime",
        "datatype": {
          "uri": "Long",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "price",
        "datatype": {
          "uri": "Double",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      }
    ],
    "uri": "bid",
    "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
  }
},
{
  "name": {
    "resourceName": "nexmark:category",
```

```
      "user": "System"
    },
    "schema": {
      "attributes": [
        {
          "sourcename": "category",
          "attributename": "id",
          "datatype": {
            "uri": "Integer",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
        {
          "sourcename": "category",
          "attributename": "name",
          "datatype": {
            "uri": "String",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
        {
          "sourcename": "category",
          "attributename": "description",
          "datatype": {
            "uri": "String",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
        {
          "sourcename": "category",
          "attributename": "parentid",
          "datatype": {
            "uri": "Integer",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        }
      ],
      "uri": "category",
      "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
    }
  },
  {
    "name": {
      "resourceName": "nexmark:person",
      "user": "System"
    },
    "schema": {
      "attributes": [
        {
          "sourcename": "person",
          "attributename": "timestamp",
          "datatype": {
            "uri": "StartTimestamp",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
```

```
      {
        "sourcename": "person",
        "attributename": "id",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "person",
        "attributename": "name",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "person",
        "attributename": "email",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "person",
        "attributename": "creditcard",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "person",
        "attributename": "city",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "person",
        "attributename": "state",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      }
    ],
    "uri": "person",
    "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
  }
},
```

```
{
  "name": {
    "resourceName": "bid",
    "user": "System"
  },
  "schema": {
    "attributes": [
      {
        "sourcename": "bid",
        "attributename": "timestamp",
        "datatype": {
          "uri": "StartTimestamp",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "auction",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "bidder",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "datetime",
        "datatype": {
          "uri": "Long",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "bid",
        "attributename": "price",
        "datatype": {
          "uri": "Double",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      }
    ],
    "uri": "bid",
    "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
  }
},
{
  "name": {
    "resourceName": "person",
    "user": "System"
```

```
},
"schema": {
  "attributes": [
    {
      "sourcename": "person",
      "attributename": "timestamp",
      "datatype": {
        "uri": "StartTimestamp",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "person",
      "attributename": "id",
      "datatype": {
        "uri": "Integer",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "person",
      "attributename": "name",
      "datatype": {
        "uri": "String",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "person",
      "attributename": "email",
      "datatype": {
        "uri": "String",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "person",
      "attributename": "creditcard",
      "datatype": {
        "uri": "String",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "person",
      "attributename": "city",
      "datatype": {
        "uri": "String",
        "type": "BASE",
        "subtype": null,
        "subSchema": null
      },
      "subschema": null
    },
    {
      "sourcename": "person",
```

```
        "attributename": "state",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      }
    ],
    "uri": "person",
    "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
  }
},
{
  "name": {
    "resourceName": "auction",
    "user": "System"
  },
  "schema": {
    "attributes": [
      {
        "sourcename": "auction",
        "attributename": "timestamp",
        "datatype": {
          "uri": "StartTimestamp",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "id",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "itemname",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "description",
        "datatype": {
          "uri": "String",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "initialbid",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
```

```
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "reserve",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "expires",
        "datatype": {
          "uri": "Long",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "seller",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      },
      {
        "sourcename": "auction",
        "attributename": "category",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
      }
    ],
    "uri": "auction",
    "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
  }
},
{
  "name": {
    "resourceName": "category",
    "user": "System"
  },
  "schema": {
    "attributes": [
      {
        "sourcename": "category",
        "attributename": "id",
        "datatype": {
          "uri": "Integer",
          "type": "BASE",
          "subtype": null,
          "subSchema": null
        },
        "subschema": null
```

```
        },
        {
          "sourcename": "category",
          "attributename": "name",
          "datatype": {
            "uri": "String",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
        {
          "sourcename": "category",
          "attributename": "description",
          "datatype": {
            "uri": "String",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        },
        {
          "sourcename": "category",
          "attributename": "parentid",
          "datatype": {
            "uri": "Integer",
            "type": "BASE",
            "subtype": null,
            "subSchema": null
          },
          "subschema": null
        }
      ],
      "uri": "category",
      "typeClass": "de.uniol.inf.is.odysseus.core.collection.Tuple"
    }
  }
]
```


Get Query Ids

Variable	Value
RESOURCE_PATH	`getQueryIds`

Retrieve a list of all query ids.

Name	Type	Explanation
token	String	From the login

Other REST interfaces

There can be more REST interfaces in Odysseus with special purposes. These interfaces are listed below:

Name	SERVICE_PATH	Explanation
Condition Monitoring	condition	Service to control the condition monitoring feature